NorthStar™

North Star Computers, Inc.
14440 Catalina St., San Leandro, CA 94577 USA
(415) 357-8500 TWX/Telex (910) 366-7001

GRAPHICS DOS
GRAPHICS BASIC
User Manual

250406

GDOS, GBASIC and ADVANTAGE are
trademarks of North Star Computers, Inc.

Copyright © 1981, by North Star Computers, Inc.
All Rights Reserved

00779A

00778A

PREFACE

This manual tells you how to use the GDOS system software.

The manual is divided into three major sections:

GETTING STARTED
: Describes the procedure and information required to make initial use of the GDOS software.

GRAPHICS DISK OPERATING SYSTEM
: Describes the GDOS and the details of its use and features.

MONITOR
: Describes the use of the North Star MONITOR program.

The NORTHSTAR GDOS Manual and the programs it describes are copyrighted by North Star Computers, Inc. All rights are reserved. No license to copy or duplicate this manual is granted with distribution or sale. You can make duplicates of North Star programs for your use only, not for sale or distribution to others.

Every effort has been made to ensure the accuracy of the material presented here. Nevertheless, experience shows that some textual errors always go undetected. If you find any errors, or have some suggestions on how to improve this manual, please contact North Star at the following address:

NORTH STAR COMPUTERS, INC.
TECHNICAL SERVICES GROUP
14440 CATALINA STREET
SAN LEANDRO, CA 94577

#250406

TABLE OF CONTENTS

1 GETTING STARTED
1.1 INTRODUCTION 1-1
1.2 DISK DRIVES AND DISKETTES 1-2
1.2.1 Insertion of Diskettes 1-2
1.2.2 Care of Diskettes 1-2
1.3 LIST OF SYSTEM SOFTWARE PROGRAMS 1-3
1.4 SYSTEM START-UP 1-4
1.5 DISKETTE BACKUP 1-5
1.5.1 Creating a Working Diskette 1-5
1.5.2 Regular Backup Procedures 1-5

2 GRAPHICS DISK OPERATING SYSTEM
2.1 INTRODUCTION 2-1
2.2 ABOUT FILES 2-2
2.2.1 Disk Addresses 2-2
2.2.2 Files 2-2
2.3 GDOS COMMANDS 2-4
2.3.1 LI - List Directory 2-4
2.3.2 CR - Create File 2-5
2.3.3 DE - Delete File 2-5
2.3.4 TY - Change File Type 2-5
2.3.5 GO - Load and Execute Program 2-5
2.3.6 JP - Execute Program 2-6
2.3.7 LF/SF - Load/Save File 2-6
2.3.8 RD/WR - Read/Write 2-6
2.3.9 IN - Initialize Diskette 2-7
2.3.10 OD - Assign Output Device 2-7
2.3.11 DO - Direct Output Mode 2-7
2.3.12 IL - Initial Load 2-7
2.4 DISK ERRORS 2-8
2.5 DISK ACCESS SUB-SYSTEM 2-9
2.5.1 Disk Library Routines 2-9
2.5.2 File Creation Using Library Routines 2-11
2.6 UTILITIES 2-12
2.6.1 DT - Disk Test 2-13
2.6.2 CF - Copy File 2-14
2.6.3 CD - Copy Disk 2-14
2.6.4 CO - Compact 2-15
2.6.5 CK - Check Diskette 2-15
2.7 CHARACTER I/O SUB-SYSTEM 2-16
2.7.1 Character I/O Library Routines 2-16
2.7.2 Use of Character I/O Sub-System Functions From Command Mode 2-19
2.7.3 Use of Character I/O Sub-System Functions From User Programs 2-19
2.8 GRAPHICS SUB-SYSTEM 2-25
2.8.1 Overview of the Graphics Sub-System 2-25

2.8.2 RECTANGLE 2-27
2.8.3 POLYGON 2-27
2.8.4 ELLIPSE 2-29
2.8.5 SPECIAL LINE 2-30
2.8.6 Other Graphic Support Routines 2-31
2.8.7 Sample Assembly Language Graphics Programs 2-37
2.9 PERSONALIZING GDOS 2-59
2.9.1 Additional GDOS Personalization 2-59
2.9.2 GDOS Entry Points and Flags 2-62
2.9.3 Personalizing a New Diskette From an Old Diskette 2-71
2.9.4 ADVANTAGE Input/Output Routines 2-73

3 MONITOR
3.1 INTRODUCTION 3-1
3.2 COMMAND FORMAT 3-2
3.3 MONITOR COMMANDS 3-4
3.3.1 CM - Compare Memory 3-4
3.3.2 FM - Fill Memory 3-4
3.3.3 MM - Move Memory 3-4
3.3.4 SM - Search Memory 3-5
3.3.5 TM - Test Memory 3-5
3.3.6 DH - Display Hexadecimal 3-6
3.3.7 DA - Display ASCII 3-6
3.3.8 DS - Display and Substitute 3-6
3.3.9 JP - Jump 3-7
3.3.10 OS - Exit to Operating System 3-8
3.3.11 IL - Initial Load 3-8
3.3.12 OD/ID - Set Output/Input Device 3-8
3.3.13 Interrupting the Monitor 3-9
3.4 HARDWARE REQUIREMENTS 3-10
3.5 PERSONALIZING THE MONITOR 3-11
3.5.1 Example 3-12

APPENDICES

A DECIMAL-ASCII-HEX-BINARY CONVERSION TABLE

B CONSOLE I/O SUB-SYSTEM FUNCTION CODES

C VALUES FOR DIRECT CURSOR ADDRESSING

D RAM ALLOCATION

1.1 INTRODUCTION

This part of the manual provides the information and procedures required to make initial use of the North Star GDOS and related software. This material should be referenced at any of the following times:

A. You are about to use GDOS software with your ADVANTAGE computer for the first time.

B. You are about to use a new release of GDOS software.

The sections that follow provide:

A. Information on the disk drives and how to use them.

B. Itemization of the software provided on the GDOS software diskette.

C. Description of the GDOS software start-up procedure.

These sections should be read carefully and the specified procedures should be followed in the order given.

1.2 DISK DRIVES AND DISKETTES

Your ADVANTAGE computer includes capability for storing large amounts of data and program information on "floppy" diskettes. There are two floppy disk drives installed in your computer.

1.2.1 Insertion of Diskettes

When you insert the diskette into the disk drive, be sure that you are holding the label edge of the diskette, and that it slides all the way in, oblong hole first, with the label facing up. The write protect notch should be at the left. After the diskette is inserted, make sure the door on the drive is locked into the closed position before you attempt to use the diskette.

1.2.2 Care of Diskettes

Diskettes are delicate and should be handled with great care. Always observe the following rules in the handling and storing of diskettes.

1. Never directly touch the magnetic surfaces of a diskette.

2. Never bend or fold a diskette.

3. Keep a diskette in its protective envelope when not in use.

4. Never expose a diskette to heat, X-ray or other radiation, magnetic fields, moisture, or dust.

1.3 LIST OF SYSTEM SOFTWARE PROGRAMS

The following programs are included on the GDOS system software diskette:

GDOS	The Disk Operating System program.
CO	Compact diskette utility program.
CD	Copy diskette utility program.
CF	Copy file utility program.
DT	Disk test utility program.
CK	Utility program for verifying the accuracy of a copy of North Star product diskettes.
BASIC	The BASIC language system program with software arithmetic.
GBASIC	The BASIC language system program with software arithmetic and graphics.
M0000	The Monitor program with origin 0000H.
MC000	The Monitor program with origin C000H.

1.4 SYSTEM START-UP

To start up GDOS on a ADVANTAGE computer:

1. Turn on the ADVANTAGE power switch.
2. When the "LOAD SYSTEM" message is displayed, insert a GDOS software diskette into drive #1.
3. Press RETURN.

This automatically starts the disk bootstrap program which turns on the disk drive and loads the GDOS into RAM from the disk after which the GDOS system herald and command prompt (#) are displayed. GDOS is now ready to use.

1.5 DISKETTE BACKUP

1.5.1 Creating a Working Diskette

Before using the GDOS software it should be copied to a diskette other than the factory supplied GDOS software diskette. This diskette, called the WORKING DISKETTE, should be the one used on a daily basis. After this procedure is finished, the factory diskette should be retired to a safe place for storage with the write protect tab still installed. If the working diskette should ever be accidentally destroyed, the factory diskette can then be used to create a new working diskette.

Load the factory GDOS software diskette with the write protect tab installed in drive #1 and a blank diskette (to become the working diskette) with no write protect tab in drive #2. Then perform the following GDOS commands:

```
#IN 2
#GO CD 1 2
```

When the CD utility asks if you want to copy both sides of the diskette you should answer: NO, since the GDOS software diskette is single-sided. The complete contents of the factory diskette are then copied onto the working diskette. Now you are done and the working diskette is ready to use.

1.5.2 Regular Backup Procedures

It is an inescapable fact that any user of a computer will make mistakes in the instructions given to the computer. Most of these mistakes can be easily corrected. However, a few can cause major loss of information stored on diskettes. For example, suppose you have spent an entire month typing a data base into your computer and it is stored on a single diskette. You now wish to initialize a new diskette and type an IN command to the GDOS. It is not until the command is completed that you realize that you forgot to insert the new diskette in the disk drive and that you have just initialized the diskette which held the results of one month's work. This kind of disaster can be avoided by faithfully following these two rules:

1. Always keep a write protect tab on a diskette unless you are about to write on the diskette.

2. Always make a backup copy of any file you have just changed in any significant way.

The copy disk and copy file utility programs make the backup procedure easy. Important files or diskettes should be stored in a more permanent way. For example, a copy of the personalized working diskette should be retired to safe storage and be recovered only if the normal working diskette is destroyed.

2.1 INTRODUCTION

The Graphics Disk Operating System GDOS permits a user to issue various "commands" for maintaining and using files on diskette. The GDOS also provides program callable functions which may be evoked from user software. These functions will primarily be of interest to users who will be developing their own system software, as opposed to those users who will primarily use application systems such as GBASIC.

The GDOS occupies 8K (2000H) bytes of RAM, including 256 bytes of RAM for input/output routines. No buffer area outside the GDOS is required for any of the GDOS commands. The origin of the GDOS is E000H.

2.2 ABOUT FILES

2.2.1 Disk Addresses

The ADVANTAGE computer is supplied with double-sided disk drives, which are capable of storing 358,400 bytes of information. Each diskette consists of 70 concentric TRACKS, 35 on each side of the diskette. Each track is subdivided into 10 SECTORS. A disk sector can holds 512 bytes of information. For purposes of discussion, a FILE BLOCK is defined to be a unit of information equal to 256 bytes. A sector, therefore, contains two file blocks. Every sector on the disk is identified by a unique DISK ADDRESS - an integer from 0 through 699. For example, sector 3 of track 27 has disk address of 273. On side A track 0 is the "outermost" track and track 34 is the "innermost" track; on side B track 35 is the "innermost track and track 69 is the "outermost" track.

2.2.2 Files

The primary GDOS function is to permit the creation, deletion and use of files on diskettes. A file is an integral number of file blocks of data and occupies sequential disk sectors. For example, a particular file might occupy disk addresses 16 through 94 on a diskette loaded in drive #2. Note that files must always begin on sector boundaries, and must always contain an even number of file blocks.

The first four sectors on each diskette contain a FILE DIRECTORY which specifies a symbolic name, base address, length, and type information for each file on that diskette. The symbolic name may be up to 8 characters long, and may include any characters except blank and comma. The length of a file may be up to 1392 blocks. A directory may contain as many as 128 entries. No two files in a directory may have the same name, but it is possible for files of the same name to be in directories of diskettes loaded simultaneously on separate drives in a multiple disk drive system.

FILE TYPES

One byte in the file directory entry for each file specifies the "type" of the file. Depending on the specific type, additional bytes in the entry may have special meaning. Only 13 of the 128 possible file types

have been assigned to date. Of particular interest are the following:

type 0 - Default type. All new files are assigned type 0 until explicitly changed.

type 1 - Machine language program. This file type identifies a machine language program (object code) that may be executed directly from the GDOS with the GO command.

type 2 - BASIC program. This type of file is used to identify a BASIC program that can be LOADed or SAVEd from BASIC.

type 3 - BASIC data file. This type of file is the standard type for data files read and written by BASIC programs.

types 32 - 63 May be assigned by the user.

FILE DIRECTORY STRUCTURE

The file directory occupies disk addresses (sectors) 0 though 3. Each block in the directory holds thirty-two 16-byte entries. The symbolic name of the entry uses the first 8 bytes of an entry. An empty entry is an entry with 8 blanks (20 H). Following the symbolic name in an entry, the disk address (2 bytes), the file size (two bytes) and the type (1 byte) follow. The last three bytes of an entry are type dependent. In particular, for a type 1 file (GO file), the two bytes following the type byte contain the go-address, and for a type 2 file (BASIC program) the byte following the type byte specifies how many file blocks of the file actually contain valid data.

File directory entry:

bytes 0-7	symbolic name of entry
bytes 8-9	disk address
bytes 10-11	number of sectors in file
byte 12	file type
bytes 13-15	type-dependent information

2.3 GDOS COMMANDS

Instructions are issued to the GDOS from the keyboard by typing COMMANDS. The command format is a 2-letter mnemonic followed by any required arguments. Arguments are separated from the command mnemonic and from each other by a single blank. A command must be terminated by a carriage return before the GDOS takes any action. If a typing error occurs during typing of a command, an at-sign (@) may be typed to permit re-typing of the command. Also, a Backspace (<X) may be typed to erase the previously typed character. If the GDOS prints a question mark ("?") in response to a command, this indicates illegal form for the command or an illegal argument value.

When a file name is required as a command argument, the disk drive number may be specified by immediately following the file name with ",1", or ",2". Otherwise, drive #1 is assumed. Some sample file names are:

ABC TEST1234,2 BASIC,1

Commands may be typed whenever the prompt character (#) appears at the left margin of the screen Device numbers may be specified for some of these commands. Device zero is the ADVANTAGE screen. Devices one through six correspond to the six possible input/output ports located on the rear of the ADVANTAGE. If a printer is attached to your ADVANTAGE, it is usually attached as device (port) number one.

2.3.1 LI - List Directory

COMMAND: LI <optional device specification> <optional drive number>

REMARKS: This command will list the entire contents of the directory on the diskette loaded in the specified drive. If no drive is specified, then drive #1 is assumed. For each file, its symbolic name, starting disk address, length, and type will be printed. For type 1 files, the go-address will also be printed. To prematurely terminate a listing, a control-C may be typed. If output to a device other than the screen is desired, then the desired output device number may be specified by typing a # character followed by the device number. The device number must correspond to a device that has been interfaced to the system in both hardware and by

adding the appropriate personalized input/output routine.

2.3.2 CR - Create File

COMMAND: CR <file name> <length> <optional start address>

REMARKS: This command will create a new file on the drive indicated by the file name. The length argument specifies the number of 256-byte blocks. If no starting address is given, then the file will start after the "last" (innermost) file currently allocated on the diskette. Otherwise, the supplied starting address will be used. The CR command will only create a file directory entry - no accessing of the file itself will be done.

2.3.3 DE - Delete File

COMMAND: DE <file name>

REMARKS: This command will delete an existing file directory entry on the indicated drive. No actual accessing of the file blocks will be done. The DE command, in conjunction with the CR command, may be used to change the length of a file on the disk. If this is done, note that the type and type-dependent information will have to be re-entered.

2.3.4 TY - Change File Type

COMMAND: TY <file name> <file type> <optional go-address>

REMARKS: This command is used to change the type of the specified file on the indicated drive. If type 1 is specified, then the third argument must be supplied to specify the "go-address".

2.3.5 GO - Load and Execute Program

COMMAND: GO <file name>

REMARKS: This command is used to load the specified file into RAM from the indicated drive and begin execution. The GO command may be used only with type 1 files. The GO command will read the entire file into RAM beginning at the go-address, and then jump to the go-address. Therefore, the first byte of the file must be the entry point of the program. The GO command sets the HL register pair to a value

which points to the remainder of the command line (any characters typed after the file name) as stored in the GDOS command buffer in memory. In this way, it is possible to send arguments to a program through the command string. The maximum length of a GDOS command line is 80 characters.

2.3.6 JP - Execute Program

COMMAND: JP <hex RAM address>

REMARKS: This command will cause the computer to jump to the specified RAM address. It provides a way of executing programs which exist in the address space of the computer. Do not confuse this command with the GO command. However, like the GO command, JP sets the HL register pair to point to the remainder of the command string.

2.3.7 LF/SF - Load/Save File

COMMAND: LF <file name> <hex RAM address>
SF <file name> <hex RAM address>

REMARKS: These commands may be used to load a file from disk to RAM or to save a file from RAM to disk. The entire contents of the file will be read to or written from the area starting with the specified RAM address.

2.3.8 RD/WR - Read/Write

COMMAND: RD <disk address> <hex RAM address> <# of blocks>
WR <disk address> <hex RAM address> <# of blocks>

REMARKS: These commands may be used to read from disk to RAM or to write from RAM to disk. The WR and RD commands should be used with great care, as typing errors can have catastrophic effects. The disk address may optionally be followed by ",1" or ",2" to indicate a particular drive. Otherwise, drive #1 is assumed. The amount of data to transfer is specified as 256-byte file blocks. Note that a method of copying one diskette to another in a single drive system would involve repeated use of the RD and WR commands.

2.3.9 IN - Initialize Diskette

COMMAND: IN <optional drive number>

REMARKS: This command is used to initialize each new diskette to be used in the system. The IN command writes each block on the specified drive with ASCII blank characters (20H). This procedure initializes the directory and also guarantees that no "hard disk error" can result from access to an uninitialized file block. The IN command takes about 15 seconds. Needless to say, one should make sure that the proper diskette is loaded before issuing the IN command. Note that an initialized diskette does not contain a copy of the GDOS. The IN command does not require any buffer area outside of the GDOS memory area.

2.3.10 OD - Assign Output Device

COMMAND: OD <optional device number>

REMARKS: This command sets the "ADEV" byte in the GDOS jump table to the screen when no device number is given or to the device number specified in the optional argument. When a device number other than zero is used, subsequent output to the screen is also sent to that device. When no device number is given, device zero is assumed, which turns this feature off.

2.3.11 DO - Direct Output Mode

COMMAND: DO <optional device number> <optional terminator>

REMARKS: This command puts GDOS in "Direct Output" mode. Subsequent input is sent directly to the specified device. If no optional terminator is specified, this command terminates when a <CR> is typed. If an optional terminator is specified, then this command terminates when the specified terminator is typed.

2.3.12 IL - Initial Load

COMMAND: IL

REMARKS: This command causes a jump to the boot prom and the ADVANTAGE attempts to load GDOS from drive #1.

2.4 DISK ERRORS

Most disk operations are tried 10 times by the GDOS before reporting failure. Upon failure, an error message of the following form is printed on the screen:

```
Tx  Dy  Sxxx
```

where x=the disk error type,
y=the drive number on which the error occurred,
zzz=the disk sector address at which the error occurred.

The error types have the following meanings:

1 SYNC BYTE NOT FOUND: Indicates badly written data on the diskette, or a diskette not properly loaded into the drive, or an attempt to read an uninitialized diskette.

2 CRC COMPARE ERROR: Indicates badly written data.

3 VERIFY COMPARE ERROR: Indicates data on disk does not compare with RAM in a verify operation.

4 NO INDEX PULSE: Indicates wrong type of diskette or badly loaded diskette.

5 SINGLE DENSITY READ OR WRITE ATTEMPTED.

6 WRITE PROTECT: Indicates a write operation was attemped to a write protected diskette.

If the GDOS prints a question-mark(?) in response to a command, this indicates illegal form for the command or an illegal argument value.

2.5 DISK ACCESS SUB-SYSTEM

This section describes how user machine language software may interface to the GDOS for the accessing of disk files.

2.5.1 Disk Library Routines

The GDOS ENTRY POINTS AND FLAGS section shows the entry points for each of the routines to be described here. The exact interfacing requirements are described in that section. The GDOS uses the stack pointer existent at call time, and some of the GDOS library routines may require as much as 30 bytes of stack storage. Note that the GDOS may be re-entered without using the bootstrap PROM. Now follows a discussion of each library routine.

DLOOK

This routine searches for a specified file name in the directory of the indicated disk drive. If the specified name begins with a blank, then an "empty" file directory entry is looked up. On failure to find the requested entry, HL is set to the value of the first free disk address on the indicated drive following the last file on the diskette. The file name must be in the correct syntax.

On success, HL contains a pointer into a buffer in GDOS RAM that has a copy of the sought entry. The pointer addresses the first byte following the symbolic name (i.e., byte 8). Also, on return, the ACC specifies the disk drive which was determined from the name passed as argument.

DWRIT

This routine is used to write back to diskette an updated file directory entry which was previously found using DLOOK. No disk activity may occur between the DLOOK and the DWRIT call.

DCOM

This routine may be used to issue an arbitrary disk read or write command. On a read request, DCOM will try 10 times for a successful read before giving up and branching to HDERR. DCOM will fail return if the supplied arguments are out of bounds. However, great care should be used to avoid calling DCOM with incorrect arguments.

GDOS

This is an entry point to the GDOS command processor. It can be used to return control to a loaded GDOS without requiring a PROM bootstrap load.

DOSERR

When a control-C is typed at the keyboard during a diskette directory listing, or when GDOS is passed a file name which is syntactically incorrect, GDOS branches to the JMP instruction stored at this location. If left unmodified, the DOSERR JMP transfers control back to a GDOS error-handling routine. Modifying the address contained in this JMP instruction will allow a user's application program to retain control under the above-named error conditions.

HDERR

HDERR branches to GDOS code that prints an error message and then enters the GDOS command processor. GDOS branches to HDERR whenever a read attempt fails despite 10 retries. For your software to retain control in the event of a hard disk error, it must modify the address of the HDERR JMP instruction (e.g., LXI H,ADDR; SHLD HDERR+1). The stack is set to the stack pointer value after the call to DCOM. HL is set to the disk address at which the error was discovered. [Note: Software for dealing with hard disk errors is notoriously difficult. It is suggested that due to the expected low frequency of hard disk errors, for most applications the existing HDERR action will be sufficient. Hard disk errors will result primarily from careless use (e.g. forgetting to initialize a diskette, or from removing a diskette while writing is in progress). Hard disk errors can also result from power failure during writing, or from a hardware system failure.]

LIST

This routine will list the file directory of the specified drive. The listing format will be exactly the same as the listing format obtained with the GDOS LI command.

OFTEN

This routine is called at least once every 40 milliseconds when DCOM has been called to perform disk operations and whenever an interrupt occurs.

OFTEN may be personalized to a routine to poll for input/output requests. The OFTEN routine may execute for up to three seconds, and disk activity will continue when the OFTEN routine returns. OFTEN must preserve all registers except the accumulator and may only use twelve bytes of stack space.

2.5.2 File Creation Using Library Routines

Here is a procedure for creating a new file using the above routines: First use DLOOK to search for the desired new name - if DLOOK succeeds then a file of that name aread[exists and should not be created. On failure, HL will have the disk address which should be used as the starting address of the new file. Next, use DLOOK to find an empty directory entry by looking up a blank name. If this call to DLOOK fails, then the directory fails. On success, use the pointer in HL to copy the new file name into the directory entry, and copy in the disk address and length and type information. Finally, call DWRIT to copy the new directory entry back to the disk. A common variation on this technique is to start using a file once the proper starting address is determined and to defer the creation of the directory entry until successful completion of the operation when the length of the file has been determined.

2.6 UTILITIES

There are FIVE operations which may be considered as part of the GDOS but are actually implemented as GO files. The operations, and their corresponding GO file names are:

DT - Disk Test.
CF - Copy File.
CD - Copy Disk.
CO - Compact disk
CK - Check the accuracy of a North Star Software product diskette

Complete descriptions of the utilities follow. Some of the arguments to the utilities can be listed on the command line where "GO" is typed. For example:

```
#GO DT 1
```

This tells the DT utility which drive is to be tested. Any arguments which you do not supply to the utility on the GO command line are explicitly requested by the utility.

The origin in memory of each of the utilities lies just at 0000H. Each of the utilities requires a 5K buffer area. The amount of RAM required by a utility may be computed by adding the buffer size to the size of the utility on diskette. Because the utilities load at the same address as the standard version of GBASIC and many other applications programs, you should be careful that no programs or data be overwritten and therefore lost as a result of using a utility.

You may wish to use a utility to operate on a diskette different than the diskette that holds the utility program. In this case, you must change diskettes after the utility has been loaded into RAM. Each of the utilities allows a different diskette to be loaded before actually beginning its operation. Diskettes can be switched any time after the utility makes its first request for input. Do not answer that request until the switch, if any, has been made!

Typical user-computer interaction is given as EXAMPLES for each of the utilities. In these examples, note that the GDOS prompt given is a pound-sign (#). In examples, the symbol <CR> comes immediately after the user's responses to indicate that a line of user input

must always be terminated by striking the RETURN key.

2.6.1 DT - Disk Test

The Disk Test utility tests the specified drive and the diskette loaded in that drive. The following cycle is continuously repeated:

a) The entire diskette is written with data, starting at sector 0. An incrementing pattern is used. If the read after write check is enabled (see GDOS section ADDITIONAL GDOS PERSONALIZATION), then each track is verified immediately after it is written.

b) The data on the entire diskette is verified, starting at sector 0. If any sector cannot be read or contains data different than what was written, an error message is printed on the screen and the test stops.

c) If no errors have been detected by this point, the message

    PASS COMPLETED.

    is printed on the screen.

To terminate a disk test, type control-C. A diskette used for a disk test does not emerge from the test containing the information which was previously on it. Also, a diskette which was used for a disk test must be initialized before it is subsequently used for data storage.

EXAMPLES

```
#GO DT<CR>
DRIVE NUMBER: 2<CR>
LOAD DISKETTE AND PRESS RETURN TO BEGIN TEST.<CR>
PASS COMPLETE.
PASS COMPLETE.
PASS COMPLETE.
CONTROL-C STOP          User types control-C here.
#

#GO DT 2 <CR>
LOAD DISKETTE AND PRESS RETURN TO BEGIN TEST.<CR>
T3 D2 S352
#
```

2.6.2 CF - Copy File

The Copy File utility copies the contents and type information from a source file to a destination file. The destination file may be a file which already exists, but if it does not, it is created automatically. If the destination file already exists it must be at least as large as the source file (in 256-byte file blocks). Whether the destination file exists or not, CF asks if the destination file should be written in double or single density. The source and destination files may be on different diskettes loaded on different drives, or they may be on the same diskette.

EXAMPLES

```
#GO CF<CR>
FROM FILE: TEST<CR>
TO FILE: PROGRAM,2<CR>
EXISTING FILE. PRESS RETURN TO BEGIN COPY.<CR>
COPY COMPLETED.
#

#GO CF ABC ABC1<CR>
NEW FILE.  PRESS RETURN TO BEGIN COPY.<CR>
COPY COMPLETED.
#
```

2.6.3 CD - Copy Disk

The Copy Disk utility copies the entire contents of a diskette loaded on one specified drive to a diskette loaded in another specified drive. After the copy is completed, the destination diskette will contain all the same information as the source diskette. If any information on the source diskette is impossible to read, the copy terminates. The copy operation can be retried after the bad sector has been rewritten.

EXAMPLE

```
#GO CD<CR>
COPY FROM DRIVE: 1<CR>
TO DRIVE: 2<CR>
LOAD DISKETTES AND PRESS RETURN TO BEGIN COPY.<CR>
COPY COMPLETED.
#
```

2.6.4 CO - Compact

The Compact utility is used to "compact" the file space on a diskette. Any unused disk space between existing files is eliminated by moving the files toward track 0. Thus, the CO utility can be used to reclaim disk space after files have been deleted or shortened, or in case files were created in such a way as to leave gaps of disk space between them.

Before actually beginning to move files on the diskette, CO checks the file directory for any "overlapping" files. Overlapping files are any files which include at least one sector in common. Overlapping files can only be created when the optional <disk address> argument is used with the GDOS CR command, or by applications programs which create such files. If any overlapping files are discovered by the CO check, the file names are printed on the screen and the operation is aborted. If overlapping files exist, the compaction may yield unpredictable results. (NOTE: The special case of a file with disk address beginning at 0 is ignored by this check, and by the compaction process.)

A compaction can take from 1 to 30 seconds.

EXAMPLES

```
#GO CO<CR>
LOAD DISKETTE AND SPECIFY DRIVE #: 1<CR>
PRESS RETURN TO COMPACT<CR>
COMPACTION COMPLETED.
#

#GO CO 2<CR>
THE FOLLOWING FILES HAVE CONFLICTS
DATA1
TEST123
#
```

2.6.5 CK - Check Diskette

The check utility generates a CRC code based on the entire contents of a diskette. When compared against the published check code for a North Star product diskette, this number indicates whether the checked diskette is an exact copy of the product as shipped by North Star without subsequent modification or personalization.

2.7 CHARACTER I/O SUBSYSTEM FUNCTIONS

GDOS is designed to be able to interface to most terminal input/output configurations. There are four routines required by the GDOS: character input (CIN), character output (COUT), control-C detect (CONTC), and terminal initialization (TINIT). The input/output routines are located in the 256 byte region from F600 to F6FF(hex).

2.7.1 Character I/O Library Routines

CIN

The purpose of CIN is to obtain a single character of input from an input device and to return the value of that character in the accumulator. When CIN is called, the accumulator will contain a device number. This value, in the range 0 to 7, specifies from which of eight possible input devices the single character of input is to be obtained. Device 0 is always assumed to be the main terminal or "console". The standard GDOS uses the ADVANTAGE keyboard as input device 0 and whatever is connected to slots 1-6 in the back as devices 1-6 respectively. Device 7 is not implemented. CIN must do a RET to the calling routine when the input character is ready in the accumulator. The accumulator is the only register which may be modified by the CIN routine. If the input routine is complex enough to require the use of other registers, their values when CIN is called must be saved, and then restored before CIN returns.

COUT

This routine sends a single character of output information to an output device. The character to be output is provided to COUT in the B-register, and the output device number is provided in the accumulator. When COUT has finished sending the output character to the appropriate device, the character itself must be in the accumulator as well as the B-register, and the routine must do a RET back to the calling routine. No registers, other than the accumulator, may be modified by the action of the COUT routine. The standard GDOS uses the ADVANTAGE display screen as output device 0, and I/O slots 1-6 as devices 1-6.

CONTC

This routine detects if a control-C has been typed on the console. No information is passed to the routine in any of the registers, and no registers need be saved or restored by CONTC. They are all available for unrestricted use by the routine. If a control-C has been typed, the routine should set the Zero flag. If no character has been typed, the Zero and Carry flags should be cleared. If a character has been typed, it must be returned in the accumulator and the Carry flag must be set. If the character was a control-C, then the Zero flag should be set. If the character typed was not a control-C, then the Zero flag should instead be cleared. As soon as the Zero and Carryflags are given their proper values, CONTC must do a RET. CONTC should not wait for a character to be typed. If no character has been typed, it should do a RET immediately after clearing the flags.

TINIT

Many terminals require a special initialization procedure to be followed immediately after they are turned on for use. For example, the interface electronics (such as the ADVANTAGE standard serial interface) may require initialization after power-on or reset. The TINIT routine is called once by the GDOS right after the bootstrap load and should contain any instructions which implement this one time initialization for all input/output devices used in your system. Since many terminals do not need to be initialized, you may not need to use TINIT. TINIT may freely use all registers, without having to save or restore any. The TINIT routine should do a RET when finished.

ISTAT AND OSTAT

These routines are status tests and are used to determine whether an input device has more data available or an output device is ready for more data. Like CIN and COUT, these routines are called with a device number in the accumulator and only the accumulator and flags may be modified. If the specified device is not implemented, then the device that would take its place must be tested instead. The status of the tested device is returned in the Zero flag (true if ready or data available, false if not), and the number

of the device actually tested is returned in the accumulator.

The ADVANTAGE character I/O subsystem displays both characters and graphics images. GDOS includes routines which cause characters and graphics images to appear on the screen, routines which move the cursor, change display modes, and modify characters input from the console keyboard. The ADVANTAGE character I/O subsystem handles the characters it receives in one of two ways:

1. Displayable characters A-Z, a-z, 0-9, !@#$%^&*()-_=+{}[]|/|:;'"<>,.?`~ appear as characters on the screen.

2. Valid control codes and escape sequences cause cursor movement, display mode changes, and changes in the way input characters are handled. Invalid control codes are ignored.

The method for accessing video functions is described in the following section. Table 2-1 gives a detailed description of each video function. Appendix B summarizes the valid control codes and escape sequences.

2.7.2 Use of Character I/O Subsystem Functions From Command Mode

The character I/O subsystem functions can be accessed directly from the GDOS command mode by using the DO command.

EXAMPLE:

The following command sequence will clear the screen:

```
DO <CR>
CTRL-D <CR>
```

2.7.3 Use of Character I/O Subsystem Functions From User Programs

The character I/O subsystem functions can be accessed by programs written in machine language by loading the code(s) into Register B, loading a 0 into Register A, and calling the DOS COUT0 routine located at F80DH.

EXAMPLES:

To clear the screen and home the cursor, the machine language routine would be:

```
06 04     CLRSC     MVI B,004H   *SET UP CTRL-D
3E 00               MVI A,000H   *SPECIFY DEVICE ZERO
CD 0D F8            CALL F80DH   *CALL COUT0 ROUTINE
   C9                  RET       *GO BACK TO MAIN ROUTINE
```

To set the cursor at position 23 on line 14, the machine language routine would be:

```
06 1B     SETCR     MVI B,01BH   *SET UP ESCAPE
3E 00               MVI A,000H   *SPECIFY DEVICE ZERO
CD 0D F8            CALL F80DH   *SEND THE ESCAPE
06 3D               MVI B,03DH   *SET UP =
3E 00               MVI A,000H   *SPECIFY DEVICE ZERO
CD 0D F8            CALL F80DH   *SEND THE =
06 37               MVI B,037H   *POSITION 23 + OFFSET
OF 32
3E 00               MVI A,000H   *SPECIFY DEVICE ZERO
CD 0D F8            CALL F80DH   *SEND THE POSITION
06 2E               MVI B,02EH   *LINE 14 + OFFSET OF 32
3E 00               MVI A,000H   *SPECIFY DEVICE ZERO
CD 0D F8            CALL F80DH   *SEND THE LINE
C9                RET            *GO BACK TO MAIN ROUTINE
```

TABLE 2-1

VIDEO CONTROL CODES

CHARACTER	DECIMAL CODE	HEX CODE	DESCRIPTION
CONTROL-A	1	01H	REVERSE VIDEO ON Presents characters as dark on a green background.
CONTROL-B	2	02H	REVERSE VIDEO OFF Presents characters as green on a dark background.
CONTROL-D	4	04H	VIDEO RESET The screen blanks, the cursor moves to the top left corner, and all parameters are reset to the default state: smooth scrolling off auto wrap-around off mask bit 7 on cursor on keyboard audio feedback off screen on reverse video off hybrid mode off
CONTROL-E	5	05H	KEYBOARD AUDIO FEEDBACK ON When audio feedback is on a "click" will be heard every time a key is struck.
CONTROL-F	6	06H	KEYBOARD AUDIO FEEDBACK OFF No sound is heard when keys are struck.
CONTROL-G	7	07H	BELL Sounds the standard bell tone.
CONTROL-H	8	08H	BACKSPACE The cursor moves one space to the left. If auto-wraparound is on and the cursor is at the beginning of the line, then a reverse linefeed occurs and the cursor moves to the end of the previous line. No action occurs if the cursor is at the

CHARACTER	DECIMAL CODE	HEX CODE	DESCRIPTION
			beginning of the line and auto-wraparound is off.
CONTROL-J	10	0AH	LINEFEED The cursor moves down one line. If the cursor is on the top line, then the screen scrolls down one line, the top line is blanked, and the cursor does not move.
CONTROL-K	11	0B	REVERSE LINE FEED The cursor moves up one line. If the cursor is on the top line, then the screen scrolls down one line, the top line is blanked, and the cursor does not move.
CONTROL-L	12	0CH	FORESPACE The cursor moves one space to the right. If auto-wraparound is on and the cursor is at the end of the line, then a linefeed occurs and the cursor moves to the beginning of the next line. No action takes place if auto-wraparound is off and the cursor is at the end of the line.
CONTROL-M	13	0DH	RETURN The cursor moves to the beginning of the current line.
CONTROL-N	14	0EH	CLEAR TO END OF LINE All characters starting at the cursor position and continuing to the end of the line are erased. The cursor does not move.
CONTROL-O	15	0FH	CLEAR TO END OF SCREEN All characters starting at the current cursor position and continuing to the end of the

VIDEO CONTROL CODES (continued)

CHARACTER	DECIMAL CODE	HEX CODE	DESCRIPTION
			screen are erased. The cursor does not move.
CONTROL-Q	17	11H	SMOOTH SCROLLING ON - FAST Turns on smooth scrolling at fast speed.
CONTROL-R	18	12H	HYBRID MODE ON The screen is blanked, the top 200 scan lines reserved for graphics, the bottom 4 lines are reserved for text and act as if they are the entire console, and the cursor is moved to the top text line. Smooth scrolling, if on, is suspended while hybrid mode is on.
CONTROL-S	19	13H	HYBRID MODE OFF Places the screen back in TEXT mode. Subsequent line feeds will scroll all 24 lines including any graphics image. If smooth scrolling was on prior to entering hybrid mode, then smooth scrolling continues.
CONTROL-T	20	14H	SMOOTH SCROLLING ON Turns on smooth scrolling at slow speed.
CONTROL-U	21	15H	SMOOTH SCROLLING OFF Subsequent linefeeds will appear to be instantaneous.
CONTROL-X	24	18H	CURSOR ON The cursor becomes visible.
CONTROL-Y	25	19H	CURSOR OFF The cursor becomes invisible. All subsequent cursor controls work exactly the same.

CHARACTER	DECIMAL CODE	HEX CODE	DESCRIPTION
CONTROL-\	28	1CH	SCREEN ON Turns the video screen on.
CONTROL-]	29	1DH	SCREEN OFF Turns the screen off. Characters on the screen are maintained in the video memory, and further output is also stored. When the screen is turned back on, all output will reappear.
CONTROL-^	30	1EH	HOME CURSOR Moves the cursor to the upper left corner.
CONTROL-_	31	1FH	NEW LINE Has the effect of a return and then a line feed.
ESC $	27 36	1BH 24H	AUTO-WRAPAROUND ON Affects operations of backspace (CONTROL-H) and forespace (CONTROL-L).
ESC %	27 37	1BH 25H	AUTO-WRAPAROUND OFF Affects operations of backspace (CONTROL-H and forespace (CONTROL-L).
ESC = X Y	27 61 X Y	1BH 3DH X Y	DIRECT CURSOR ADDRESSING The cursor moves to line X, column Y. Note that the values of X and Y must be the actual desired value plus an offset of 32. Appendix 3 may be used to determine the proper values for X and Y. EXAMPLE: To move the cursor to line 14, column 23, output the following decimal values: 27 61 46 55

CHARACTER	DECIMAL CODE	HEX CODE	DESCRIPTION
DELETE	127	7FH	DESTRUCTIVE BACKSPACE The cursor is moved back one character space and the character at the new cursor position is erased.

2.8 GRAPHICS SUB-SYSTEM

2.8.1 Overview of the Graphics Sub-System

The graphics sub-system includes two sets of functions: geometric routines, and graphic support routines. The four geometric routines, POLYGON, RECTANGLE, ELLIPSE, and SPECIAL LINE enable you to draw points, lines, and a wide range of two-dimensional figures. The three graphic support routines, CLEAR, BLOCK, and CHAR enable you to, respectively, clear the screen and scroll the screen to bring line 0 to the top, transfer a rectangular array of data between RAM and the screen, and display characters on the screen.

To incorporate the following graphic routines in machine language programs, set the registers as described below, and call OF721H.

GEOMETRIC FUNCTIONS: GENERAL

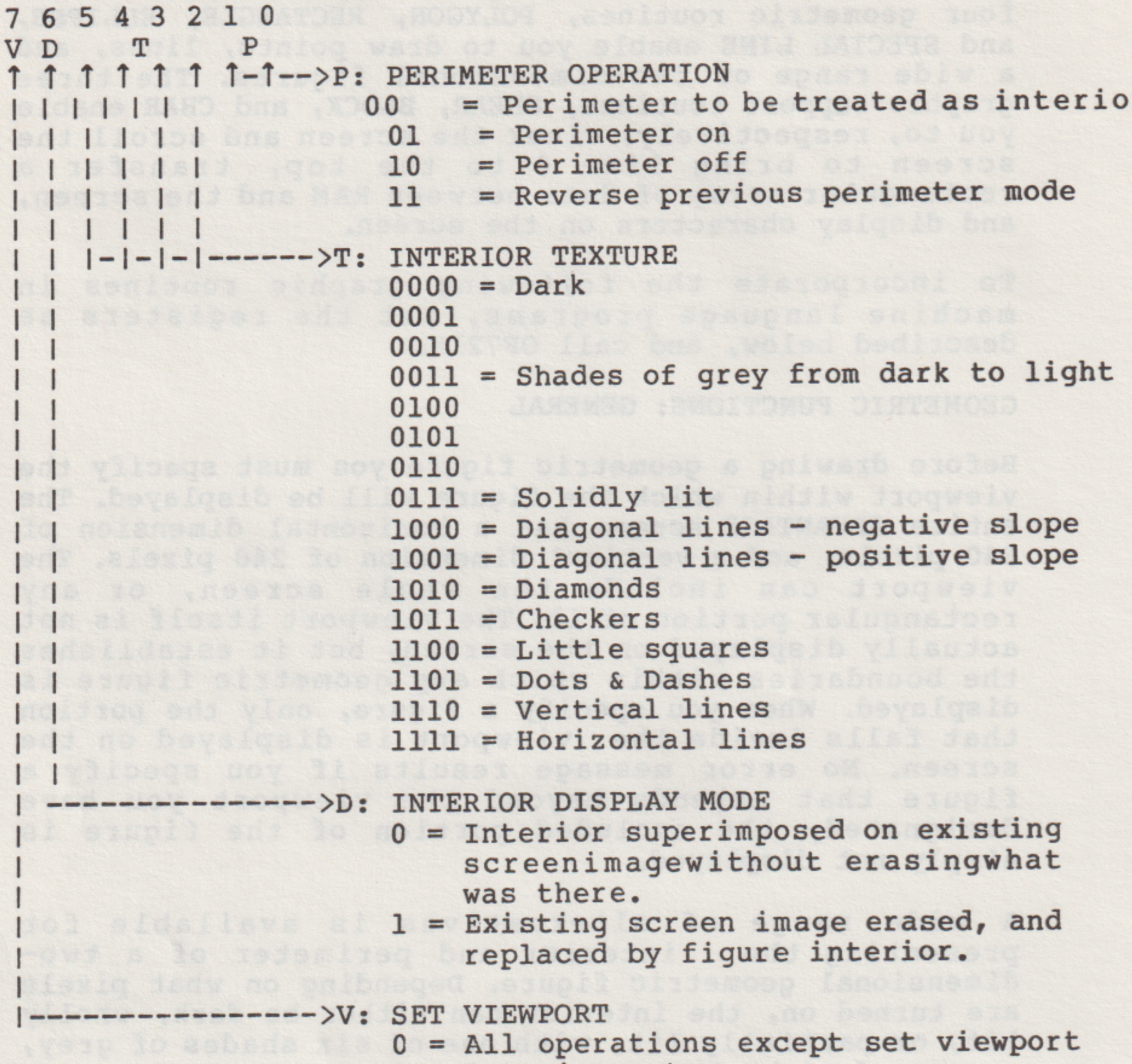
Before drawing a geometric figure you must specify the viewport within which the figure will be displayed. The entire ADVANTAGE screen has a horizontal dimension of 640 pixels, and a vertical dimension of 240 pixels. The viewport can include the whole screen, or any rectangular portion of it. The viewport itself is not actually displayed on the screen, but it establishes the boundaries within which any geometric figure is displayed. When you specify a figure, only the portion that falls inside the viewport is displayed on the screen. No error message results if you specify a figure that extends beyond the viewport you have designated; the excluded portion of the figure is simply not displayed.

A wide range of alternatives is available for presenting the interior and perimeter of a two-dimensional geometric figure. Depending on what pixels are turned on, the interior can either be dark, wholly lit, or partially lit, with one of six shades of grey, or one of eight textures.

The interior can replace the current screen display, or be superimposed on it. The perimeter can appear as a solid line around the interior, be shaded like the interior, or be omitted. It is also possible to reverse the mode of the previous image, e.g., turn off, or omit, a perimeter that was shown as a solid line in the previous screen display.

All four geometric routines use the following command byte structure to determine how the figure will be displayed. This command byte is loaded in register C.

COMMAND BYTE STRUCTURE


7 6 5 4 3 2 1 0
V D T P
P: PERIMETER OPERATION
00 = Perimeter to be treated as interior
01 = Perimeter on
10 = Perimeter off
11 = Reverse previous perimeter mode
T: INTERIOR TEXTURE
0000 = Dark
0001
0010
0011 = Shades of grey from dark to light
0100
0101
0110
0111 = Solidly lit
1000 = Diagonal lines - negative slope
1001 = Diagonal lines - positive slope
1010 = Diamonds
1011 = Checkers
1100 = Little squares
1101 = Dots & Dashes
1110 = Vertical lines
1111 = Horizontal lines
D: INTERIOR DISPLAY MODE
0 = Interior superimposed on existing screenimagewithout erasingwhat was there.
1 = Existing screen image erased, and replaced by figure interior.
V: SET VIEWPORT
0 = All operations except set viewport
1 = Set viewport

Note that when the command byte is 0, nothing happens.

A final general consideration applies to all four geometric routines. The HL register is loaded with the RAM start address of a table that contains the coordinates of: the vertices of a polygon, the opposite

corners of a rectangle, the center of an ellipse, or the end point of a special line.

The coordinate pair(s) X,Y consists of positive or negative two's complement 16-bit values. The coordinate values can lie anywhere between the decimal values -32,768 and +32,767, but only points whose coordinates fall with the viewport are displayed on the screen.

2.8.2 RECTANGLE

The RECTANGLE function serves two purposes: to set the viewport, and to draw rectangles. In both cases the sides must be parallel to the horizontal and vertical axes, X and Y, and the two points you designate are the opposite corners of the shape. It is possible to set the viewport and draw a rectangle in a single step.

To set the viewport, specify a rectangle that does not exceed the size of the screen. If the input parameters specify a rectangle that extends off the screen, the previous viewport will not be changed and error code 01H will be given.

Input Parameters:

Register	Description
A	Set to a value of 2 - for RECTANGLE routine.
C	See command byte explanation. Set bit 7 to 1 to set the viewport. Set bit 7 to 0 to draw rectangle. Set bit 7 to 1 to set viewport and draw rectangle simultaneously.
HL	RAM start address of 8-byte table containing coordinates of two points. Stored as X1, Y1, X2, Y2, these specify opposite corners of rectangle.

2.8.3 POLYGON

The polygon routine draws points and lines as well as many-sided figures. This routine only guarantees success with convex figures. If you specify a concave figure, the graphics sub-system will attempt to draw it, and return error code 02H if the effort fails.

However, many concave polygons can be drawn, as well as figures that are not well-defined polygons at all.

A polygon is well-defined if it has one contiguous interior and exterior. A set of ten random points, for example, will probably not form a well-defined polygon when connected in the order given, since the interior may be several separate regions. This is not an error, and shading of the interior will still occur in a well-defined manner. The POLYGON routine fills in the polygon one horizontal scan line at a time, from top to bottom. At each scan line it finds all places where an edge intersects the scan line, pairs them off in order from left to right, and fills between each pair. To accomplish this, the program uses a variable-length list of 'active edges.' Sometimes the complexity of the shape of the polygon may get so great that the list of active edges gets too long to fit in the program's internal buffer. When this happens the extra buffer feature must be used (see Input Parameters). 600H bytes is the maximum that will ever be needed to handle polygons of 64 sides or less. Some examples of polygons that can be drawn without using the extra buffer feature are:

1) Any 18 or fewer points whether or not they determine a well-defined polygon,

2) 64 or fewer points defining a convex polygon.

Input Parameters:

Register	Description
A	Set to a value of 1 for POLYGON routine
B	Number of vertices: 1 = a point 2 = a line 3 - 100 = a polygon
C	Command Byte (plus, if bit 7 set, use extra buffer)
HL	RAM start address of table containing coordinates designating each individual point, or each endpoint for a line, or each vertex for a polygon. The table follows the form:

	X1, Y1,...XN, YN. Vertices must be specified in order clockwise or counterclockwise.
IX	Starting address of extra buffer
IY	Length of extra buffer

NOTE: IX, IY are ignored if bit 7 of C is off.

2.8.4 ELLIPSE

The ELLIPSE routine draws ellipses and circles, arcs, pie slices, and areas enclosed by an arc and a subtended chord.

The greatest and the least diameters are drawn parallel to the axes. This routine cannot be used to draw a rotated ellipse.

For this routine, the vertical radius displayed on the screen has the value, i.e., the number of pixels, specified in register E. The horizontal radius displayed on the screen is twice the value, i.e., number of pixels, specified in register D. This doubling of the specified horizontal radius makes it possible to draw a circle by setting the horizontal and vertical radii equal, even though the horizontal pixel density is twice the vertical pixel density.

If either radius is zero, nothing is drawn.

Input Parameters:

Register	Description
A	Set to a value of 3 for ELLIPSE routine. Set to a value of: 0 = whole ellipse or circle; in this case IX and IY are ignored. 1 = arc 2 = pie slice 3 = area enclosed by arc and subtended chord
C	See command byte explanation.
D	Set to half value of desired

	horizontal radius. Value is doubled in resulting figure.
E	Set to value of desired vertical radius
HL	RAM start address of 4-byte table containing coordinates of center point. Stored as X, Y.
IX	Start angle (0-360). Ignore for whole ellipse or circle.
IY	Stop angle (0-360). Ignore for whole ellipse or circle.

If IX or IY is greater than 360, the value mod 360 is used.

2.8.5 SPECIAL LINE

The SPECIAL LINE routine provides a second method of drawing a line. The POLYGON routine allows you to draw a line by specifying the start and end points. The SPECIAL LINE routine draws a line with only the end point specified. The start point is supplied by the end point of a previous line, drawn with either the POLYGON or SPECIAL LINE routine. If the last graphic command was a geometric routine that drew something other than a line, the line drawn with a SPECIAL LINE routine may start from an unpredictable location. The non-geometric graphic routines described below, CLEAR, BLOCK and CHAR can be interspersed between SPECIAL LINE calls without losing track of where the next SPECIAL LINE should start.

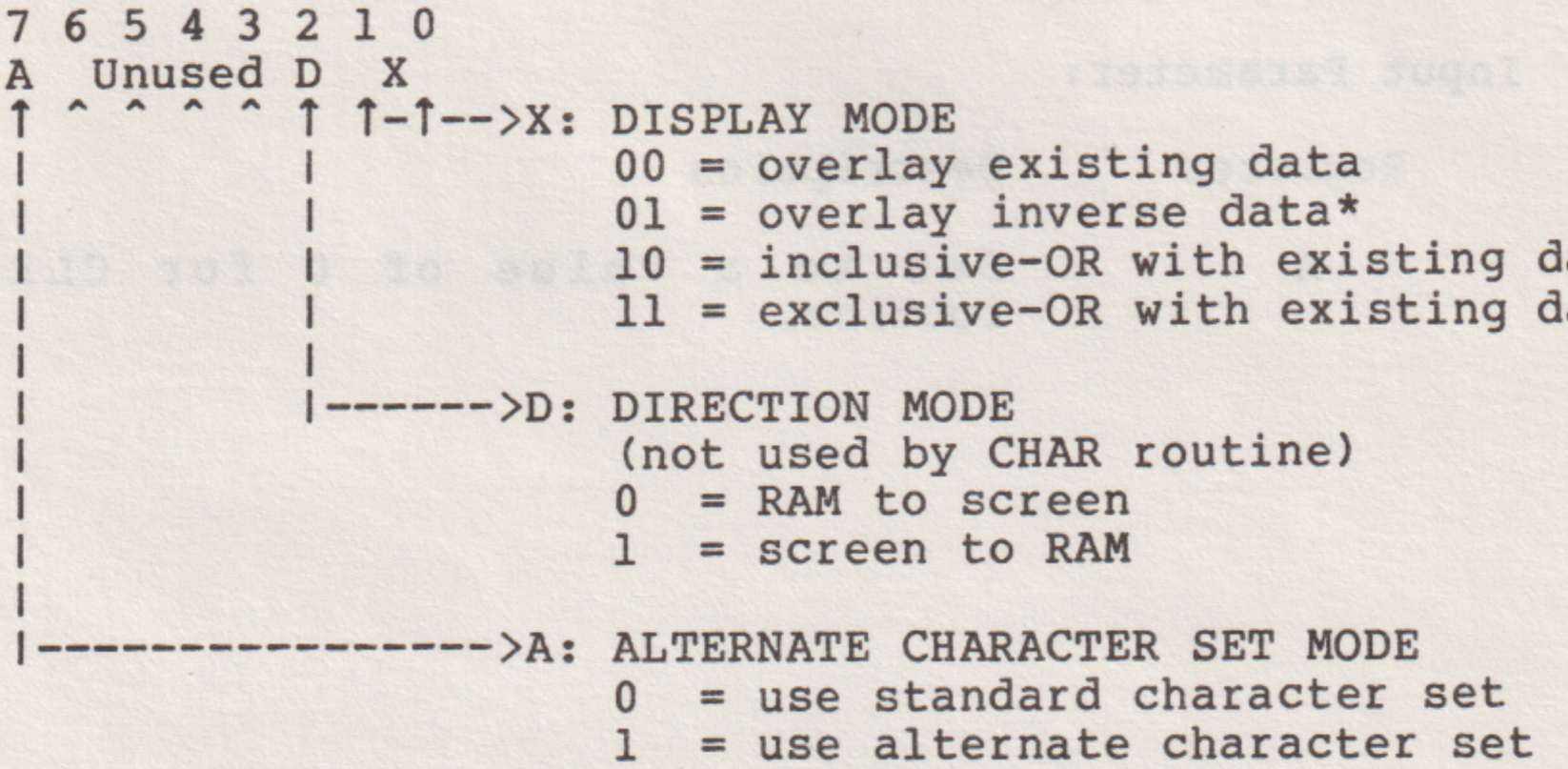
Input Parameters:

Register	Description
A	Set the value to 6 for SPECIAL LINE routine.
C	See command byte explanation.
DE	Set the value to the Y coordinate of end point.
HL	Set the value to the X coordinate of end point.

2.8.6 Other Graphic Support Routines

The three additional graphic routines, CLEAR, BLOCK, and CHAR, are distinct from the geometric routines both in the capabilities they provide, and the structure of the command byte used in register C.

The CLEAR routine does not use the command byte. The BLOCK and CHAR routines use the same command byte structure:


7 6 5 4 3 2 1 0
A Unused D X
X: DISPLAY MODE
00 = overlay existing data
01 = overlay inverse data*
10 = inclusive-OR with existing data**
11 = exclusive-OR with existing data***
D: DIRECTION MODE
(not used by CHAR routine)
0 = RAM to screen
1 = screen to RAM
A: ALTERNATE CHARACTER SET MODE
0 = use standard character set
1 = use alternate character set

* Each bit that is turned on in the RAM data is turned off on the screen, etc.

** The data in RAM is superimposed on the screen image, without erasing the previous display.

*** If an individual bit in the RAM data is turned on, the corresponding bit on the screen display is reversed. If the individual bit in RAM is turned off, it does not affect the screen display.

CLEAR

The CLEAR routine clears the screen and scrolls it to line 0. In other words, if you had previously used a character I/O function to scroll the screen up so that line 10 appeared on the top of the screen, the CLEAR routine would restore the screen to its default setting, with line 0 positioned at the top of the screen.

This routine does not use the command byte at all.

Input Parameter:

Register	Description
A	Set to a value of 0 for CLEAR routine.

BLOCK

The BLOCK routine makes it possible to move a rectangular array of data between main RAM and the screen. The data transferred is always a multiple of 8 bits. The data in RAM is stored starting from the upper left byte down to the lower left byte, then continuing down one column after another, to finish at the lower right byte.

The data can be placed in a block anywhere on the screen. The position of the block is determined by the coordinates of the upper left hand corner of the block.

BLOCK uses the entire display memory, the 240 visible lines of pixels, as well as the 16 invisible lines at the bottom of the screen. If the data reaches the bottom of the screen, the transfer operation will write on the invisible lines, then wrap around to the top of the screen.

Input Parameters:

Register	Description
A	Set to a value of 4 for BLOCK routine
C	See command byte explanation
D	Set to width of block in 8-bit columns
E	Set to height of block in lines
HL	RAM start address of block
IX	X coordinate of upper left corner of block in screen display
IY	Y coordinate of upper left corner of block in screen display in lower 8 bits of IY. The upper 8 bits of IY are ignored.

The transfer of data between RAM and screen occurs with exceptional speed when all the following conditions are met: C=0 (overlay RAM to screen), HL > 8000H (data array stored in upper two quadrants), and IX is an exact multiple of 8 (block aligned on byte boundaries).

CHAR

The CHAR routine draws characters on the screen. There are three major ways in which the graphic sub-system CHAR routine differs from the way characters are displayed by the I/O sub-system. First, the I/O sub-system allows you to display only the standard character set, whereas the graphic CHAR routine allows you to specify an alternate character set, in addition to the standard set. Second, the cursor is an essential feature of the character I/O sub-system. Each character is placed at the current cursor position. The graphic CHAR routine does not employ a cursor. Instead, the location of each character is determined by the coordinates of the upper left corner of the character.

Third, the character I/O sub-system requires that each character be displayed within the standard grid, that can hold 80 characters across, and 24 lines of characters down. Since the graphic CHAR routine addresses individual pixels, it give you much greater freedom in terms of positioning characters. The top of a character can be placed at any of the 256 lines of pixels in the display memory and a character can be drawn across a byte boundary. This allows proportional spacing, for example.

If you specify a non-printable ASCII character, error code 03H results. If the coordinates that position a character are off the screen, error code 04H results.

Input Parameters

Register	Description
A	Set the value to 5 for CHAR routine.
B	Code of printable ASCII character.
C	See command byte explanation. Set bit 7 to 0 for standard character set, to 1 for alternate char. set.
E	Y coordinate of upper left corner of character.
HL	X coordinate of upper left corner of character.

IX RAM start address of table containing alternate character set pixel data.

CONSTRUCTING AN ALTERNATE CHARACTER SET

An alternate character set consists of 95 7-byte entries which represent the displayable ASCII character set. This table starts with the representation for the ASCII SPACE (20H) and ends with the representation for the ASCII TILDE (7EH).

The 7-byte entry for a character specifies a bit map on a pixel-by-pixel basis. Bits 7 and 6 of byte 1 specify the presence or absence of descenders; if bit 7=1 then the character contains 1 descender; if both bits 7 and 6 = 1 then the character contains 2 descenders. Bits 5, 4, 3, 2, 1 and 0 of all 7 bytes, represent pixels in an 8 by 10 grid. Characters are usually displayed as a 5 by 7 matrix within the 8 by 10 grid; the largest character which can be displayed is a 6 by 7 matrix within the 8 by 10 grid.

The character grid below illustrates the 8 by 10 grid and the 6 by 7 maximum character matrix, and equates the 7 byte entry to the grid.

	COL7 BIT7	COL6 BIT6	COL5 BIT5	COL4 BIT4	COL3 BIT3	COL2 BIT2	COL1 BIT1	COL0 BIT0
ROW 0								
ROW 1								
BYTE 1	*	**	X	X	X	X	X	X
ROW 2								
BYTE 2			X	X	X	X	X	X
ROW 3								
BYTE 3			X	X	X	X	X	X
ROW 4								
BYTE 4			X	X	X	X	X	X
ROW 5								
BYTE 5			X	X	X	X	X	X
ROW 6								
BYTE 6			X	X	X	X	X	X
ROW 7								
BYTE 7			X	X	X	X	X	X
ROW 8								
DESCENDER								
ROW 9								
DESCENDER								

HOW DESCENDERS ARE DISPLAYED

If bit 7 of byte 1 represented in the grid by a * is 1, the character matrix will begin in ROW2 and will end in ROW8. If bits 7 and 6 of byte 1 represented in the grid by * and ** are 1, the character matrix will begin in ROW3 and will end in ROW9. Row 0 is always blank. The value of Bits 7 and 6 of bytes 2, 3, 4, 5, 6, and 7 have no effect on the displayed character.

2.8.7 Sample Assembly Language Graphics Programs

SET VIEWPORT - Sample Program

The following assembly language program will set the viewport equal to the size of the ADVANTAGE video screen. Although the RECTANGLE routine is used to set the viewport, no output appears on the screen if the command byte specified in the C Register is 80H.

```
                0010 ;SET VIEWPORT
8000            0020        ORG  08000H
F721 =          0030 GRAF   EQU  0F721H ;GRAPHICS ENTRY POINT
F80D =          0040 COUT   EQU  0F80DH ;OUTPUT ENTRY POINT
F810 =          0050 CINP   EQU  0F810H ;INPUT ENTRY POINT
8000 3E02       0060 SETVP  MVI  A,002H ;IDENTIFY RECTANGLE
8002 0E80       0070        MVI  C,080H ;VIEWPORT SETUP CODE
8004 212F80     0080        LXI  H,VP   ;VIEWPORT COORDINATES
8007 CD21F7     0090 DOIT   CALL GRAF   ;CALL GRAPHICS MANAGER
800A 323780     0100        STA  ERR    ;SAVE ERROR CODE
800D 3E00       0110 WAIT   MVI  A,000H ;SPECIFY KEYBOARD
800F CD10F8     0120        CALL CINP   ;INPUT CHARACTER
8012 FE00       0130        CPI  000H   ;CHECK FOR INPUT
8014 CA0D80     0140        JZ   WAIT   ;LOOP IF NONE
8017 3A3780     0150 DONE   LDA  ERR    ;GET ERROR CODE
801A FE00       0160        CPI  000H   ;WAS THERE AN ERROR?
801C CA2E80     0170        JZ   NEXT   ;DO NEXT ROUTINE IF NOT
801F FEFF       0180        CPI  0FFH   ;WAS ERROR CODE FFH?
8021 C22680     0190        JNZ  DONE2  ;SKIP IF NOT
8024 3E05       0200        MVI  A,005H ;CHANGE TO CODE 5
8026 C630       0210 DONE2  ADI  030H   ;MAKE IT AN ASCII DIGIT
8028 47         0220        MOV  B,A    ;PREPARE OUTPUT
8029 3E00       0230        MVI  A,000H ;TO SCREEN
802B CD0DF8     0240        CALL COUT   ;SEND IT
802E C9         0250 NEXT   RET         ;RETURN TO CALLING ROUTINE
802F 0000       6000 VP     DW   0000H  ;X1
8031 0000       6010        DW   0000H  ;Y1
8033 7F02       6020        DW   027FH  ;X2
8035 F000       6030        DW   00EFH  ;Y2
8037 00         6040 ERR    DB   000H   ;LAST ERROR CODE
```

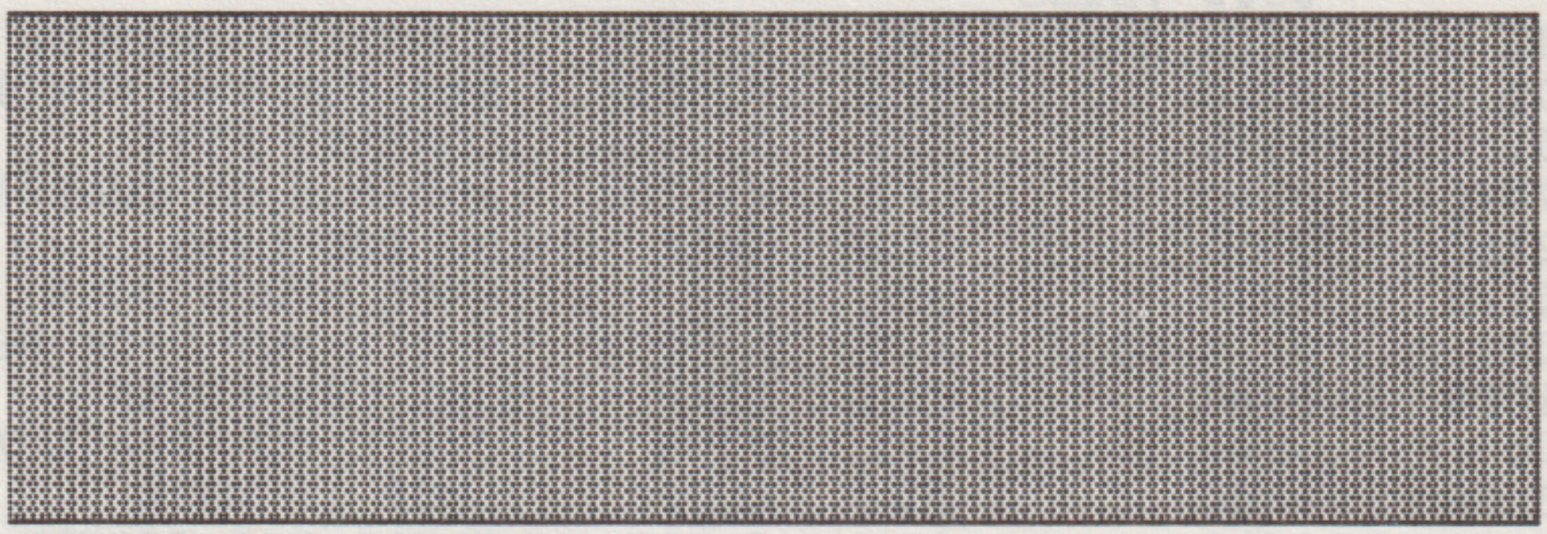
RECTANGLE - Sample Program

The following assembly language program will draw a rectangle on the ADVANTAGE video screen. Note that this routine first sets the viewport, then draws the rectangle. This is done because the viewport is set to enclose the entire ADVANTAGE video screen, while the rectangle is drawn on only a portion of the screen. If the viewport and rectangle size were identical, the viewport could be set at the same time that the rectangle was drawn by adding 80H to command byte specified in the rectangle portion of the program.

```
                0010          ;SET VIEWPORT
8000            0020          ORG   08000H
F721 =          0030 GRAF     EQU   0F721H  ;GRAPHICS ENTRY POINT
F80D =          0040 COUT     EQU   0F80DH  ;OUTPUT ENTRY POINT
F810 =          0050 CINP     EQU   0F810H  ;INPUT ENTRY POINT
8000 3E02       0060 SETVP    MVI   A,002H  ;IDENTIFY RECTANGLE
8002 0E80       0070          MVI   C,080H  ;VIEWPORT SETUP CODE
8004 216580     0080          LXI   H,VP    ;VIEWPORT COORDINATES
8007 CD21F7     0090 DOIT     CALL  GRAF    ;CALL GRAPHICS MANAGER
800A 326D80     0100          STA   ERR     ;SAVE ERROR CODE
800D 3E00       0110 WAIT     MVI   A,000H  ;SPECIFY KEYBOARD
800F CD10F8     0120          CALL  CINP    ;INPUT CHARACTER
8012 FE00       0130          CPI   000H    ;CHECK FOR INPUT
8014 CA0D80     0140          JZ    WAIT    ;LOOP IF NONE
8017 3A6D80     0150 DONE     LDA   ERR     ;GET ERROR CODE
801A FE00       0160          CPI   000H    ;WAS THERE AN ERROR?
801C CA2F80     0170          JZ    NEXT    ;DO NEXT ROUTINE IF NOT
801F FEFF       0180          CPI   0FFH    ;WAS ERROR CODE FFH?
8021 C22680     0190          JNZ   DONE2   ;SKIP IF NOT
8024 3E05       0200          MVI   A,005H  ;CHANGE TO CODE 5
8026 C630       0210 DONE2    ADI   030H    ;MAKE IT AN ASCII DIGIT
8028 0607       0220          MVI   B,A     ;PREPARE OUTPUT
802A 3E00       0230          MVI   A,000H  ;TO SCREEN
802C CD0DF8     0240          CALL  COUT    ;SEND IT
                0300 ;RECTANGLE
802F 3E02       0310 NEXT     MVI   A,002H  ;IDENTIFY RECTANGLE ROUTINE
8031 0E51       0330          MVI   C,051H  ;COMMAND BYTE
8033 216E80     0340          LXI   H,RECT  ;ADDRESS OF COORDINATES
8036 CD21F7     0350          CALL  GRAF    ;DO THE LINE
8039 326D80     0360          STA   ERR     ;SAVE ERROR CODE
803C 3E00       0370 LOOP     MVI   A,000H  ;SET INPUT FROM KEYBOARD
803E CD10F8     0380          CALL  CINP    ;GET CHARACTER
8041 FE00       0390          CPI   000H    ;CHECK FOR CHARACTER
8043 CA3C80     0400          JZ    LOOP    ;LOOP IF NONE
8046 CD5F80     0410          CALL  CLEAR   ;CLEAR THE SCREEN
8049 3A6D80     0420          LDA   ERR     ;GET ERROR CODE BACK
804C FE00       0430          CPI   000H    ;WAS THERE AN ERROR?
```

```
804E C8          0440          RZ              ;IF NOT, RETURN
804F FEFF        0450          CPI   0FFH      ;WAS IT FFH?
8051 C25680      0460          JNZ   LOOP2     ;IF NO, SKIP
8054 3E05        0470          MVI   A,005H    ;CHANGE IT TO A 5
8056 C630        0480 LOOP2    ADI   030H      ;MAKE IT AN ASCII DIGIT
8058 47          0490          MOV   B,A       ;SET UP OUTPUT
8059 3E00        0500          MVI   A,000H    ;TO SCREEN
805B CD0DF8      0510          CALL  COUT      ;SEND IT
805E C9          0520          RET             ;RETURN
805F 3E00        0700 CLEAR    MVI   A,000H    ;CLEAR SCREEN CODE
8061 CD21F7      0710          CALL  GRAF      ;SEND IT
8064 C9          0720          RET             ;GO BACK
8065 0000        6000 VP       DW    0000H     ;X1
8067 0000        6010          DW    0000H     ;Y1
8069 7F02        6020          DW    027FH     ;X2
806B F000        6030          DW    00F0H     ;Y2
806D 00          6040 ERR      DB    000H      ;LAST ERROR CODE
806E 5000        7000 RECT     DW    0050H     ;X1
8070 5000        7010          DW    0050H     ;Y1
8072 3002        7020          DW    0230H     ;X2
8074 A000        7030          DW    00A0H     ;Y2
```

PICTURE OF THE RECTANGLE DRAWN BY THE PRECEDING PROGRAM



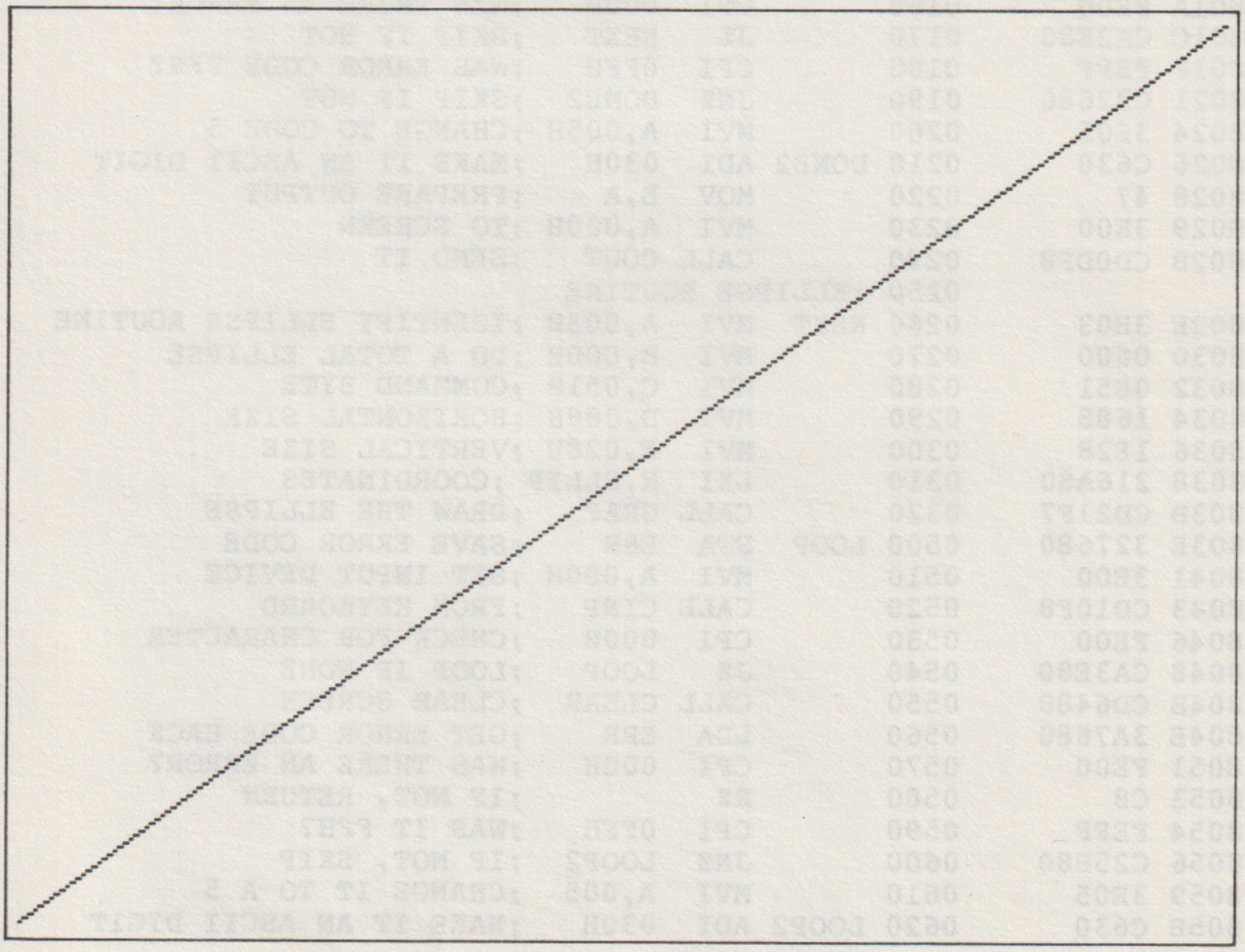
LINE - Sample Program

The following assembly language program draws a line on the ADVANTAGE video screen. Note that the POLYGON routine is used to draw the line.

```
                0010          ;SET VIEWPORT
8000            0020          ORG   08000H
F721 =          0030 GRAF     EQU   0F721H  ;GRAPHICS ENTRY POINT
F80D =          0040 COUT     EQU   0F80DH  ;OUTPUT ENTRY POINT
F810 =          0050 CINP     EQU   0F810H  ;INPUT ENTRY POINT
8000 3E02       0060 SETVP    MVI   A,002H  ;IDENTIFY RECTANGLE
8002 0E80       0070          MVI   C,080H  ;VIEWPORT SETUP CODE
8004 216780     0080          LXI   H,VP    ;VIEWPORT COORDINATES
8007 CD21F7     0090 DOIT     CALL  GRAF    ;CALL GRAPHICS MANAGER
800A 326F80     0100          STA   ERR     ;SAVE ERROR CODE
800D 3E00       0110 WAIT     MVI   A,000H  ;SPECIFY KEYBOARD
800F CD10F8     0120          CALL  CINP    ;INPUT CHARACTER
8012 FE00       0130          CPI   000H    ;CHECK FOR INPUT
8014 CA0D80     0140          JZ    WAIT    ;LOOP IF NONE
8017 3A6F80     0150 DONE     LDA   ERR     ;GET ERROR CODE
801A FE00       0160          CPI   000H    ;WAS THERE AN ERROR?
801C CA2F80     0170          JZ    NEXT    ;DO NEXT ROUTINE IF NOT
801F FEFF       0180          CPI   0FFH    ;WAS ERROR CODE FFH?
8021 C22680     0190          JNZ   DONE2   ;SKIP IF NOT
8024 3E05       0200          MVI   A,005H  ;CHANGE TO CODE 5
8026 C630       0210 DONE2    ADI   030H    ;MAKE IT AN ASCII DIGIT
8028 0607       0220          MVI   B,A     ;PREPARE OUTPUT
802A 3E00       0230          MVI   A,000H  ;TO SCREEN
802C CD0DF8     0240          CALL  COUT    ;SEND IT
                0300 ;LINE
802F 3E01       0310 NEXT     MVI   A,001H  ;IDENTIFY POLYGON ROUTINE
8031 0602       0320          MVI   B,002H  ;SET FOR TWO VERTICES
8033 0E51       0330          MVI   C,051H  ;COMMAND BYTE
8035 217080     0340          LXI   H,LINE  ;ADDRESS OF COORDINATES
8038 CD21F7     0350          CALL  GRAF    ;DO THE LINE
803B 326F80     0360          STA   ERR     ;SAVE ERROR CODE
803E 3E00       0370 LOOP     MVI   A,000H  ;SET INPUT FROM KEYBOARD
8040 CD10F8     0380          CALL  CINP    ;GET CHARACTER
8043 FE00       0390          CPI   000H    ;CHECK FOR CHARACTER
8045 CA3E80     0400          JZ    LOOP    ;LOOP IF NONE
8048 CD6180     0410          CALL  CLEAR   ;CLEAR THE SCREEN
804B 3A6F80     0420          LDA   ERR     ;GET ERROR CODE BACK
804E FE00       0430          CPI   000H    ;WAS THERE AN ERROR?
8050 C8         0440          RZ            ;IF NOT, RETURN
8051 FEFF       0450          CPI   0FFH    ;WAS IT FFH?
8053 C25880     0460          JNZ   LOOP2   ;IF NO, SKIP
8056 3E05       0470          MVI   A,005H  ;CHANGE IT TO A 5
8058 C630       0480 LOOP2    ADI   030H    ;MAKE IT AN ASCII DIGIT
805A 47         0490          MOV   B,A     ;SET UP OUTPUT
```

```
805B 3E00      0500          MVI   A,000H  ;TO SCREEN
805D CD0DF8    0510          CALL  COUT    ;SEND IT
8060 C9        0520          RET           ;RETURN
8061 3E00      0700 CLEAR    MVI   A,000H  ;CLEAR SCREEN CODE
8063 CD21F7    0710          CALL  GRAF    ;SEND IT
8066 C9        0720          RET           ;GO BACK
8067 0000      6000 VP       DW    0000H   ;X1
8069 0000      6010          DW    0000H   ;Y1
806B 7F02      6020          DW    027FH   ;X2
806D F000      6030          DW    00EFH   ;Y2
806F 00        6040 ERR      DB    000H    ;LAST ERROR CODE
8070 0000      7000 LINE     DW    0000H   ;X1
8072 EF00      7010          DW    00EFH   ;Y1
8074 7F02      7020          DW    027FH   ;X2
8076 0000      7030          DW    0000H   ;Y2
```

PICTURE OF THE LINE DRAWN BY THE PRECEDING PROGRAM



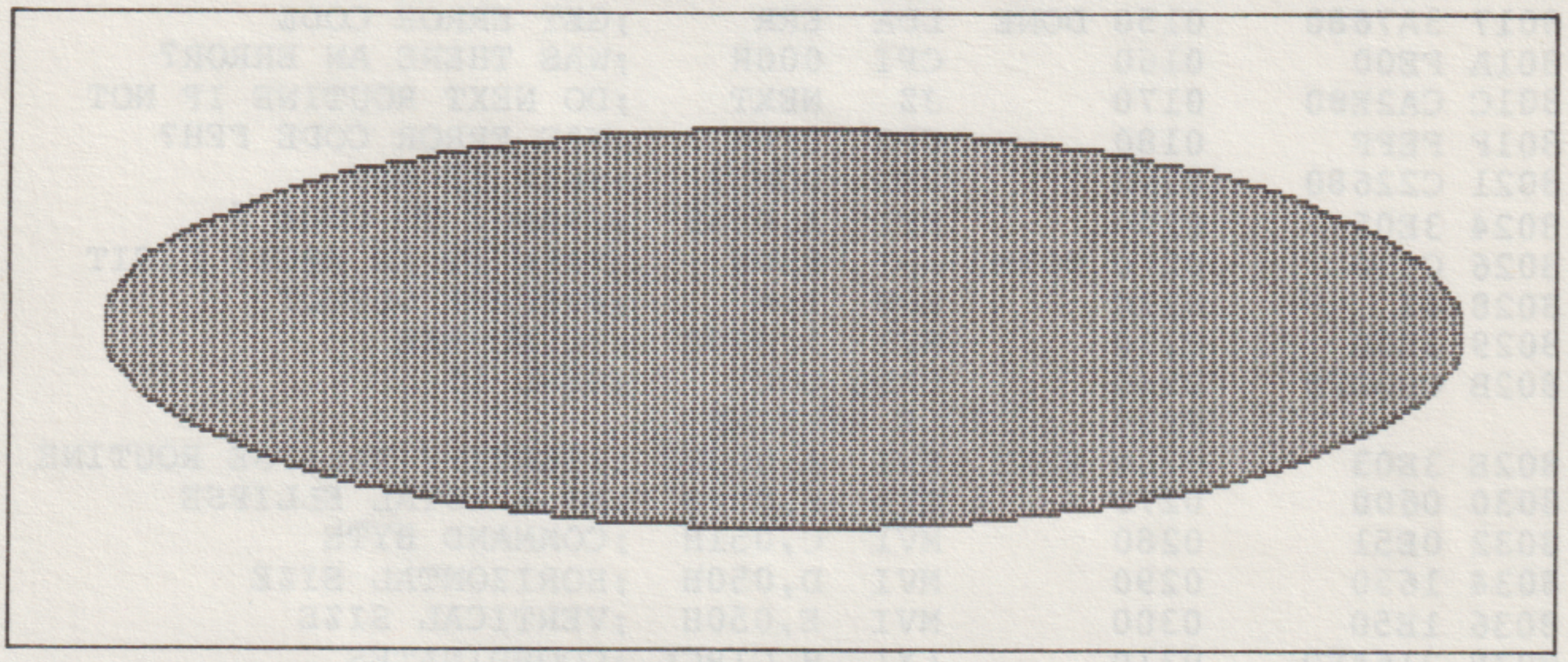
ELLIPSE - Sample Program

The following program draws an ellipse on the ADVANTAGE video screen.

```
                0010 ;SET VIEWPORT
8000            0020       ORG  08000H
F721 =          0030 GRAF  EQU  0F721H ;GRAPHICS ENTRY POINT
F80D =          0040 COUT  EQU  0F80DH ;OUTPUT ENTRY POINT
F810 =          0050 CINP  EQU  0F810H ;INPUT ENTRY POINT
8000 3E02       0060 SETVP MVI  A,002H ;IDENTIFY RECTANGLE
8002 0E80       0070       MVI  C,080H ;VIEWPORT SETUP CODE
8004 216E80     0080       LXI  H,VP   ;VIEWPORT COORDINATES
8007 CD21F7     0090 DOIT  CALL GRAF   ;CALL GRAPHICS MANAGER
800A 327680     0100       STA  ERR    ;SAVE ERROR CODE
800D 3E00       0110 WAIT  MVI  A,000H ;SPECIFY KEYBOARD
800F CD10F8     0120       CALL CINP   ;INPUT CHARACTER
8012 FE00       0130       CPI  000H   ;CHECK FOR INPUT
8014 CA0D80     0140       JZ   WAIT   ;LOOP IF NONE
8017 3A7680     0150 DONE  LDA  ERR    ;GET ERROR CODE
801A FE00       0160       CPI  000H   ;WAS THERE AN ERROR?
801C CA2E80     0170       JZ   NEXT   ;SKIP IF NOT
801F FEFF       0180       CPI  0FFH   ;WAS ERROR CODE FFH?
8021 C22680     0190       JNZ  DONE2  ;SKIP IF NOT
8024 3E05       0200       MVI  A,005H ;CHANGE TO CODE 5
8026 C630       0210 DONE2 ADI  030H   ;MAKE IT AN ASCII DIGIT
8028 47         0220       MOV  B,A    ;PREPARE OUTPUT
8029 3E00       0230       MVI  A,000H ;TO SCREEN
802B CD0DF8     0240       CALL COUT   ;SEND IT
                0250 ;ELLIPSE ROUTINE
802E 3E03       0260 NEXT  MVI  A,003H ;IDENTIFY ELLIPSE ROUTINE
8030 0600       0270       MVI  B,000H ;DO A TOTAL ELLIPSE
8032 0E51       0280       MVI  C,051H ;COMMAND BYTE
8034 1688       0290       MVI  D,088H ;HORIZONTAL SIZE
8036 1E28       0300       MVI  E,028H ;VERTICAL SIZE
8038 216A80     0310       LXI  H,ELLIP ;COORDINATES
803B CD21F7     0320       CALL GRAF   ;DRAW THE ELLIPSE
803E 327680     0500 LOOP  STA  ERR    ;SAVE ERROR CODE
8041 3E00       0510       MVI  A,000H ;SET INPUT DEVICE
8043 CD10F8     0520       CALL CINP   ;FROM KEYBOARD
8046 FE00       0530       CPI  000H   ;CHECK FOR CHARACTER
8048 CA3E80     0540       JZ   LOOP   ;LOOP IF NONE
804B CD6480     0550       CALL CLEAR  ;CLEAR SCREEN
804E 3A7680     0560       LDA  ERR    ;GET ERROR CODE BACK
8051 FE00       0570       CPI  000H   ;WAS THERE AN ERROR?
8053 C8         0580       RZ          ;IF NOT, RETURN
8054 FEFF       0590       CPI  0FFH   ;WAS IT FFH?
8056 C25B80     0600       JNZ  LOOP2  ;IF NOT, SKIP
8059 3E05       0610       MVI  A,005  ;CHANGE IT TO A 5
805B C630       0620 LOOP2 ADI  030H   ;MAKE IT AN ASCII DIGIT
```

```
805D 47        0630        MOV  B,A     ;SET UP OUTPUT
805E 3E00      0640        MVI  A,000H  ;TO SCREEN
8060 CD0DF8    0650        CALL COUT    ;SEND IT
8063 C9        0660        RET          ;RETURN TO CALLING ROUTINE
8064 3E00      0700 CLEAR  MVI  A,000H  ;CLEAR SCREEN CODE
8066 CD21F7    0710        CALL GRAF    ;SEND THE REQUEST
8069 C9        0720        RET          ;GO BACK
806A 3F01      5000 ELLIP  DW   013FH   ;X1
806C 7700      5010        DW   0077H   ;Y1
806E 0000      6000 VP     DW   0000H   ;X1
806F 0000      6010        DW   0000H   ;Y1
8072 7F02      6020        DW   027FH   ;X2
8074 F000      6030        DW   00EFH   ;Y2
8076 00        6040 ERR    DB   000H    ;LAST ERROR CODE
```

PICTURE OF THE ELLIPSE DRAWN BY THE PRECEDING PROGRAM



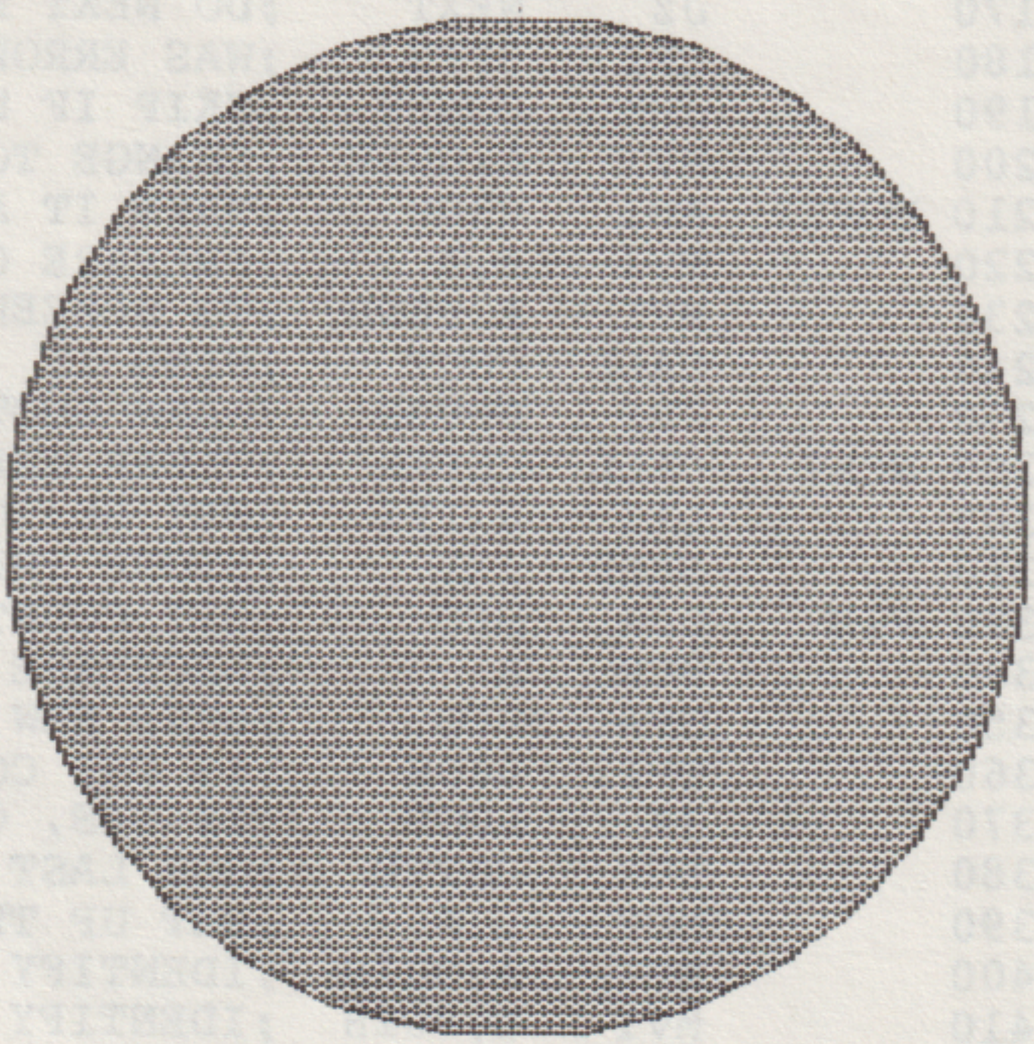
CIRCLE - Sample Program

The following assembly language program draws a circle on the ADVANTAGE video screen. The ELLIPSE routine is used to draw the circle.

```
                0010 ;SET VIEWPORT
8000            0020       ORG  08000H
F721 =          0030 GRAF  EQU  0F721H  ;GRAPHICS ENTRY POINT
F80D =          0040 COUT  EQU  0F80DH  ;OUTPUT ENTRY POINT
F810 =          0050 CINP  EQU  0F810H  ;INPUT ENTRY POINT
8000 3E02       0060 SETVP MVI  A,002H  ;IDENTIFY RECTANGLE
8002 0E80       0070       MVI  C,080H  ;VIEWPORT SETUP CODE
8004 216E80     0080       LXI  H,VP    ;VIEWPORT COORDINATES
8007 CD21F7     0090 DOIT  CALL GRAF    ;CALL GRAPHICS MANAGER
800A 327680     0100       STA  ERR     ;SAVE ERROR CODE
800D 3E00       0110 WAIT  MVI  A,000H  ;SPECIFY KEYBOARD
800F CD10F8     0120       CALL CINP    ;INPUT CHARACTER
8012 FE00       0130       CPI  000H    ;CHECK FOR INPUT
8014 CA0D80     0140       JZ   WAIT    ;LOOP IF NONE
8017 3A7680     0150 DONE  LDA  ERR     ;GET ERROR CODE
801A FE00       0160       CPI  000H    ;WAS THERE AN ERROR?
801C CA2E80     0170       JZ   NEXT    ;DO NEXT ROUTINE IF NOT
801F FEFF       0180       CPI  0FFH    ;WAS ERROR CODE FFH?
8021 C22680     0190       JNZ  DONE2   ;SKIP IF NOT
8024 3E05       0200       MVI  A,005H  ;CHANGE TO CODE 5
8026 C630       0210 DONE2 ADI  030H    ;MAKE IT AN ASCII DIGIT
8028 47         0220       MOV  B,A     ;PREPARE OUTPUT
8029 3E00       0230       MVI  A,000H  ;TO SCREEN
802B CD0DF8     0240       CALL COUT    ;SEND IT
                0250  ;CIRCLE ROUTINE
802E 3E03       0260 NEXT  MVI  A,003H  ;IDENTIFY ELLIPSE ROUTINE
8030 0600       0270       MVI  B,000H  ;DO A TOTAL ELLIPSE
8032 0E51       0280       MVI  C,051H  ;COMMAND BYTE
8034 1650       0290       MVI  D,050H  ;HORIZONTAL SIZE
8036 1E50       0300       MVI  E,050H  ;VERTICAL SIZE
8038 216A80     0310       LXI  H,CIRCL ;COORDINATES
803B CD21F7     0320       CALL GRAF    ;DRAW THE ELLIPSE
803E 327680     0500 LOOP  STA  ERR     ;SAVE ERROR CODE
8041 3E00       0510       MVI  A,000H  ;SET INPUT DEVICE
8043 CD10F8     0520       CALL CINP    ;FROM KEYBOARD
8046 FE00       0530       CPI  000H    ;CHECK FOR CHARACTER
8048 CA3E80     0540       JZ   LOOP    ;LOOP IF NONE
804B CD6480     0550       CALL CLEAR   ;CLEAR SCREEN
804E 3A7680     0560       LDA  ERR     ;GET ERROR CODE BACK
8051 FE00       0570       CPI  000H    ;WAS THERE AN ERROR?
8053 C8         0580       RZ           ;IF NOT, RETURN
8054 FEFF       0590       CPI  0FFH    ;WAS IT FFH?
8056 C25B80     0600       JNZ  LOOP2   ;IF NOT, SKIP
8059 3E05       0610       MVI  A,005   ;CHANGE IT TO A 5
```

```
805B C630       0620 LOOP2 ADI  030H    ;MAKE IT AN ASCII DIGIT
805D 47         0630       MOV  B,A     ;SET UP OUTPUT
805E 3E00       0640       MVI  A,000H  ;TO SCREEN
8060 CD0DF8     0650       CALL COUT    ;SEND IT
8063 C9         0660       RET          ;RETURN TO CALLING ROUTINE
8064 3E00       0700 CLEAR MVI  A,000H  ;CLEAR SCREEN CODE
8066 CD21F7     0710       CALL GRAF    ;SEND THE REQUEST
8069 C9         0720       RET          ;GO BACK
806A 3F01       5000 CIRCL DW   013FH    ;X1
806C 7700       5010       DW   0077H    ;Y1
806E 0000       6000 VP    DW   0000H    ;X1
806F 0000       6010       DW   0000H    ;Y1
8072 7F02       6020       DW   027FH    ;X2
8074 F000       6030       DW   00EFH    ;Y2
8076 00         6040 ERR   DB   000H     ;LAST ERROR CODE
```

PICTURE OF THE CIRCLE DRAWN BY THE PRECEDING PROGRAM



PIE CHART - Sample Program

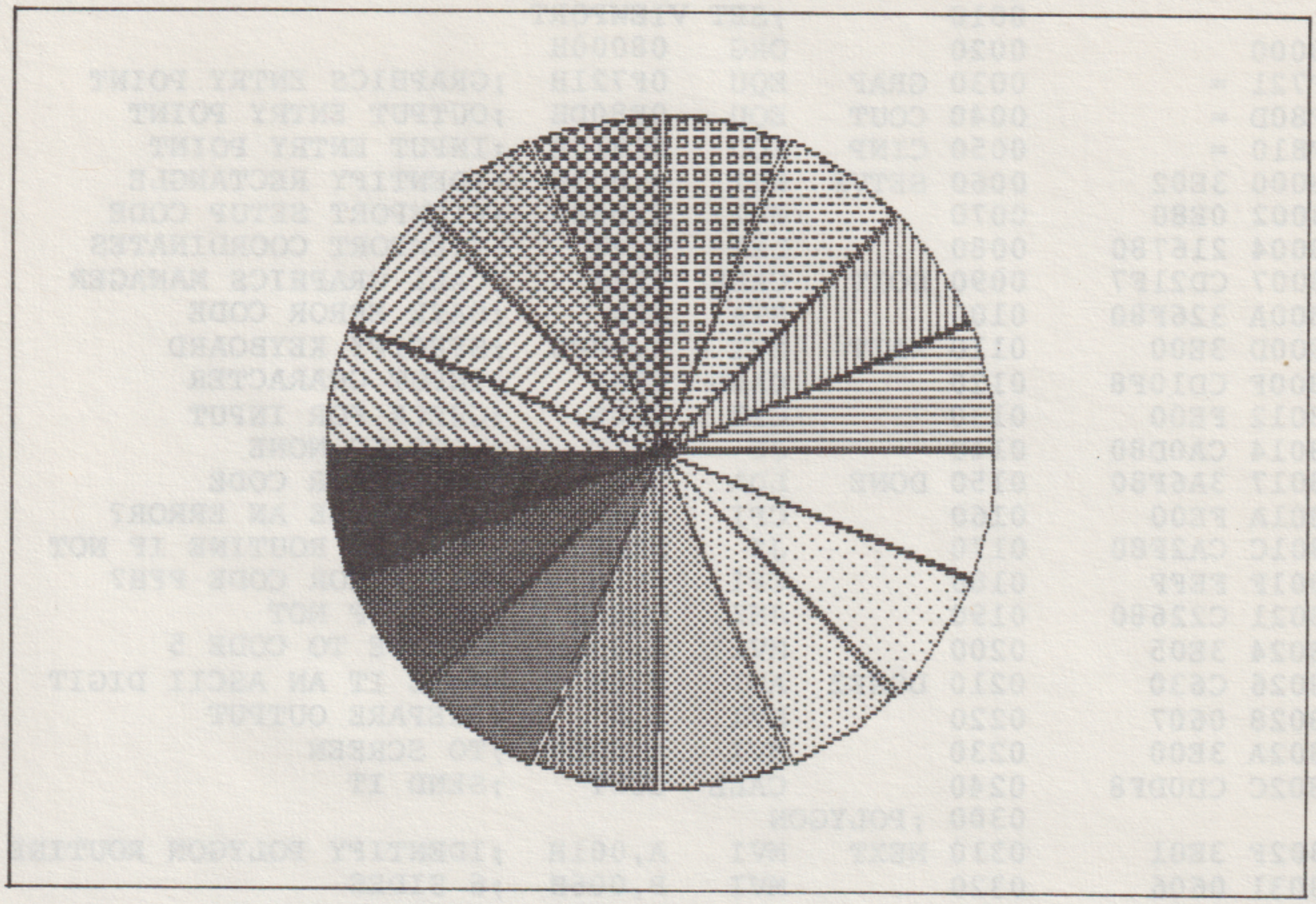
The following assembly language program draws a pie chart on the ADVANTAGE video screen. The ELLIPSE routine is used to draw the pie chart.

```
                0010          ;SET VIEWPORT
8000            0020          ORG   08000H
F721 =          0030 GRAF     EQU   0F721H  ;GRAPHICS ENTRY POINT
F80D =          0040 COUT     EQU   0F80DH  ;OUTPUT ENTRY POINT
F810 =          0050 CINP     EQU   0F810H  ;INPUT ENTRY POINT
8000 3E02       0060 SETVP    MVI   A,002H  ;IDENTIFY RECTANGLE
8002 0E80       0070          MVI   C,080H  ;VIEWPORT SETUP CODE
8004 21AD80     0080          LXI   H,VP    ;VIEWPORT COORDINATES
8007 CD21F7     0090 DOIT     CALL  GRAF    ;CALL GRAPHICS MANAGER
800A 32B980     0100          STA   ERR     ;SAVE ERROR CODE
800D 3E00       0110 WAIT     MVI   A,000H  ;SPECIFY KEYBOARD
800F CD10F8     0120          CALL  CINP    ;INPUT CHARACTER
8012 FE00       0130          CPI   000H    ;CHECK FOR INPUT
8014 CA0D80     0140          JZ    WAIT    ;LOOP IF NONE
8017 3AB980     0150 DONE     LDA   ERR     ;GET ERROR CODE
801A FE00       0160          CPI   000H    ;WAS THERE AN ERROR?
801C CA3280     0170          JZ    NEXT    ;DO NEXT ROUTINE IF NOT
801F FEFF       0180          CPI   0FFH    ;WAS ERROR CODE FFH?
8021 C22680     0190          JNZ   DONE2   ;SKIP IF NOT
8024 3E05       0200          MVI   A,005H  ;CHANGE TO CODE 5
8026 C630       0210 DONE2    ADI   030H    ;MAKE IT AN ASCII DIGIT
8028 0607       0220          MVI   B,A     ;PREPARE OUTPUT
802A 3E00       0230          MVI   A,000H  ;TO SCREEN
802C CD0DF8     0240          CALL  COUT    ;SEND IT
802F C30000     0250          JMP   0000H   ;WARM BOOT
8032 CDA780     0300 NEXT     CALL  CLEAR   ;CLEAR THE SCREEN
8035 21BE80     0305          LXI   H,IX1   ;SET HL TO COORDINATES
8038 22BC80     0310          SHLD  IXY     ;STORE THE LOCATION
803B 3ABA80     0320 PIE      LDA   SEQ     ;GET COUNTER FOR SEGMENTS
803E 3D         0330          DCR   A       ;DECREASE COUNT BY ONE
803F 32BA80     0350          STA   SEQ     ;SAVE NEW COUNTER VALUE
8042 FE00       0360          CPI   000H    ;IS THE COUNTER ZERO?
8044 CA8080     0370          JZ    LOOP    ;IF YES, GO TO LOOP
8047 3ABB80     0380          LDA   CBYTE   ;GET LAST COMMAND BYTE
804A 4F         0390          MOV   C,A     ;SET UP THE C REGISTER
804B 3E03       0400          MVI   A,003H  ;IDENTIFY ELLIPSE
804D 0602       0410          MVI   B,002H  ;IDENTIFY PIE SLICE
804F 2ABC80     0420          LHLD  IXY     ;HL GETS COORD ADDRESS
8052 5E         0430          MOV   E,M     ;DE GETS COORDINATES
8053 23         0440          INX   H
8054 56         0450          MOV   D,M
8055 23         0460          INX   H
8056 D5         0470          PUSH  D       ;SAVE COORDINATES TO
8057 FDE1       0480          POP   IY      ;PUT IN THE IY REGISTER
```

```
8059 5E        0490          MOV   E,M       ;REPEAT FOR IX REGISTER
805A 23        0500          INX   H
805B 56        0510          MOV   D,M
805C 23        0520          INX   H
805D 22BC80    0530          SHLD  IXY       ;SAVE ADDRESS AGAIN
8060 D5        0540          PUSH  D         ;SAVE COORDINATES
8061 DDE1      0550          POP   IX        ;TO PUT IN IX REGISTER
8063 1650      0560          MVI   D,050H    ;HORIZONTAL RADIUS OF PIE
8065 1E50      0570          MVI   E,050H    ;VERTICAL RADIUS OF PIE
8068 21B580    0575          LXI   H,ELLIP   ;GET COORDINATES FOR CENTER
806A CD21F7    0580          CALL  GRAF      ;DO THE GRAF NOW
806D 32B980    0590          STA   ERR       ;SAVE THE ERROR CODE
8070 FE00      0600          CPI   000H      ;WAS THERE AN ERROR?
8072 C28080    0610          JNZ   LOOP      ;EXIT TO LOOP IF YES
8075 3ABB80    0620          LDA   CBYTE     ;GET THE COMMAND BYTE
8078 C604      0630          ADI   004H      ;SET FOR NEXT TEXTURE
807A 32BB80    0640          STA   CBYTE     ;SAVE THE COMMAND BYTE
807D C33B80    0650          JMP   PIE       ;REPEAT FOR 16 SLICES
8080 3E00      0660 LOOP     MVI   A,000H    ;SET INPUT
8082 CD10F8    0670          CALL  CINP      ;FROM KEYBOARD
8085 FE00      0680          CPI   000H      ;CHECK FOR CHARACTER
8087 CA8080    0690          JZ    LOOP      ;LOOP IF NONE
808A CDA780    0700          CALL  CLEAR     ;CLEAR THE SCREEN
808D 3AB980    0710          LDA   ERR       ;GET ERROR CODE BACK
8090 FE00      0720          CPI   000H      ;WAS THERE AN ERROR?
8092 CA0000    0730          JZ    0000H     ;IF NOT, WARM BOOT
8095 FEFF      0740          CPI   0FFH      ;WAS IT CODE FFH?
8097 C29C80    0750          JNZ   LOOP2     ;SKIP IF NOT
809A 3E05      0760          MVI   A,005H    ;CHANGE IT TO A CODE 5
809C C630      0770 LOOP2    ADI   030H      ;MAKE IT AN ASCII DIGIT
809E 47        0780          MOV   B,A       ;SET UP OUTPUT
809F 3E00      0790          MVI   A,000H    ;TO SCREEN
80A1 CD0DF8    0800          CALL  COUT      ;SEND IT
80A4 C30000    0810          JMP   0000H     ;WARM BOOT
80A7 3E00      0820 CLEAR    MVI   A,000H    ;SET FOR CLEAR SCREEN
80A9 CD21F7    0830          CALL  GRAF      ;CLEAR IT NOW
80AC C9        0840          RET             ;GO BACK
80AD 0000      0850 VP       DW    0000H     ;X1
80AF 0000      0860          DW    0000H     ;Y1
80B1 7F02      0870          DW    027FH     ;X2
80B3 EF00      0880          DW    00EFH     ;Y2
80B5 3F01      0900 ELLIP    DW    013FH     ;X1
80B7 7700      0910          DW    0077H     ;Y1
80B9 00        0920 ERR      DB    000H      ;LAST ERROR CODE
80BA 11        0930 SEQ      DB    011H      ;SEQUENCE COUNTER
80BB 41        0940 CBYTE    DB    041H      ;LAST COMMAND BYTE
80BC 0000      0950 IXY      DW    0000H     ;ADDRESS OF LAST COORD
80BE 6801      0960 IX1      DW    0168H     ;THE REST ARE THE
80C0 5201      0970          DW    0152H     ;COORDINATES FOR THE
80C2 5101      0980          DW    0151H     ;PIE CHART
```

```
80C4 3B01       0990          DW    013BH
80C6 3A01       1000          DW    013AH
80C8 2501       1010          DW    0125H
80CA 2401       1020          DW    0124H
80CC 0E01       1030          DW    010EH
80CE 0D01       1040          DW    010DH
80D0 F800       1050          DW    00F8H
80D2 F700       1060          DW    00F7H
80D4 E100       1070          DW    00E1H
80D6 E000       1080          DW    00E0H
80D8 CB00       1090          DW    00CBH
80DA CA00       1100          DW    00CAH
80DC B400       1110          DW    00B4H
80DE B300       1120          DW    00B3H
80E0 9E00       1130          DW    009EH
80E2 9D00       1140          DW    009DH
80E4 8700       1150          DW    0087H
80E6 8600       1160          DW    0086H
80E8 7100       1170          DW    0071H
80EA 7000       1180          DW    0070H
80EC 5A00       1190          DW    005AH
80EE 5900       1200          DW    0059H
80F0 4400       1210          DW    0044H
80F2 4300       1220          DW    0043H
80F4 2D00       1230          DW    002DH
80F6 2C00       1240          DW    002CH
80F8 1700       1250          DW    0017H
80FA 1600       1260          DW    0016H
80FC 0000       1270          DW    0000H
80FE            1280  END     EQU   $
```

PICTURE OF THE PIE CHART DRAWN BY THE PRECEDING PROGRAM



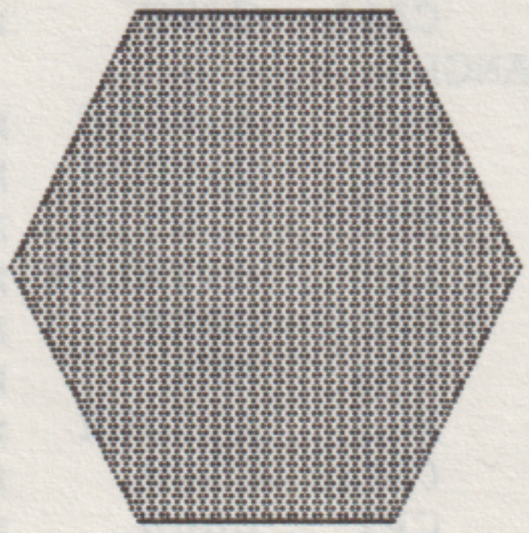
POLYGON - Sample Program

The following assembly language program draws a polygon on the ADVANTAGE video screen.

```
                0010          ;SET VIEWPORT
8000            0020          ORG   08000H
F721 =          0030 GRAF     EQU   0F721H  ;GRAPHICS ENTRY POINT
F80D =          0040 COUT     EQU   0F80DH  ;OUTPUT ENTRY POINT
F810 =          0050 CINP     EQU   0F810H  ;INPUT ENTRY POINT
8000 3E02       0060 SETVP    MVI   A,002H  ;IDENTIFY RECTANGLE
8002 0E80       0070          MVI   C,080H  ;VIEWPORT SETUP CODE
8004 216780     0080          LXI   H,VP    ;VIEWPORT COORDINATES
8007 CD21F7     0090 DOIT     CALL  GRAF    ;CALL GRAPHICS MANAGER
800A 326F80     0100          STA   ERR     ;SAVE ERROR CODE
800D 3E00       0110 WAIT     MVI   A,000H  ;SPECIFY KEYBOARD
800F CD10F8     0120          CALL  CINP    ;INPUT CHARACTER
8012 FE00       0130          CPI   000H    ;CHECK FOR INPUT
8014 CA0D80     0140          JZ    WAIT    ;LOOP IF NONE
8017 3A6F80     0150 DONE     LDA   ERR     ;GET ERROR CODE
801A FE00       0160          CPI   000H    ;WAS THERE AN ERROR?
801C CA2F80     0170          JZ    NEXT    ;DO NEXT ROUTINE IF NOT
801F FEFF       0180          CPI   0FFH    ;WAS ERROR CODE FFH?
8021 C22680     0190          JNZ   DONE2   ;SKIP IF NOT
8024 3E05       0200          MVI   A,005H  ;CHANGE TO CODE 5
8026 C630       0210 DONE2    ADI   030H    ;MAKE IT AN ASCII DIGIT
8028 0607       0220          MVI   B,A     ;PREPARE OUTPUT
802A 3E00       0230          MVI   A,000H  ;TO SCREEN
802C CD0DF8     0240          CALL  COUT    ;SEND IT
                0300 ;POLYGON
802F 3E01       0310 NEXT     MVI   A,001H  ;IDENTIFY POLYGON ROUTINE
8031 0606       0320          MVI   B,006H  ;6 SIDES
8033 0E51       0330          MVI   C,051H  ;COMMAND BYTE
8035 217080     0340          LXI   H,POLYG ;ADDRESS OF COORDINATES
8038 CD21F7     0350          CALL  GRAF    ;DO THE LINE
803B 326F80     0360          STA   ERR     ;SAVE ERROR CODE
803E 3E00       0370 LOOP     MVI   A,000H  ;SET INPUT FROM KEYBOARD
8040 CD10F8     0380          CALL  CINP    ;GET CHARACTER
8043 FE00       0390          CPI   000H    ;CHECK FOR CHARACTER
8045 CA3E80     0400          JZ    LOOP    ;LOOP IF NONE
8048 CD6180     0410          CALL  CLEAR   ;CLEAR THE SCREEN
804B 3A6F80     0420          LDA   ERR     ;GET ERROR CODE BACK
804E FE00       0430          CPI   000H    ;WAS THERE AN ERROR?
8050 C8         0440          RZ            ;IF NOT, RETURN
8051 FEFF       0450          CPI   0FFH    ;WAS IT FFH?
8053 C25880     0460          JNZ   LOOP2   ;IF NO, SKIP
8056 3E05       0470          MVI   A,005H  ;CHANGE IT TO A 5
8058 C630       0480 LOOP2    ADI   030H    ;MAKE IT AN ASCII DIGIT
805A 47         0490          MOV   B,A     ;SET UP OUTPUT
805B 3E00       0500          MVI   A,000H  ;TO SCREEN
```

```
805D CD0DF8     0510          CALL  COUT     ;SEND IT
8060 C9         0520          RET            ;RETURN
8061 3E00       0700 CLEAR    MVI   A,000H   ;CLEAR SCREEN CODE
8063 CD21F7     0710          CALL  GRAF     ;SEND IT
8066 C9         0720          RET            ;GO BACK
8067 0000       6000 VP       DW    0000H    ;X1
8069 0000       6010          DW    0000H    ;Y1
806B 7F02       6020          DW    027FH    ;X2
806D F000       6030          DW    00EFH    ;Y2
806F 00         6040 ERR      DB    000H     ;LAST ERROR CODE
8070 1701       7000 POLYG    DW    0117H    ;X1
8072 4F00       7010          DW    004FH    ;Y1
8074 6701       7020          DW    0167H    ;X2
8076 4F00       7030          DW    004FH    ;Y2
8078 8F01       7040          DW    018FH    ;X3
807A 7700       7050          DW    0077H    ;Y3
807C 6701       7060          DW    0167H    ;X4
807E 9F00       7070          DW    009FH    ;Y4
8080 1701       7080          DW    0117H    ;X5
8082 9F00       7090          DW    009FH    ;Y5
8084 EF00       7100          DW    00EFH    ;X6
8086 7700       7110          DW    0077H    ;Y6
```

PICTURE OF THE POLYGON DRAWN BY THE PRECEDING PROGRAM



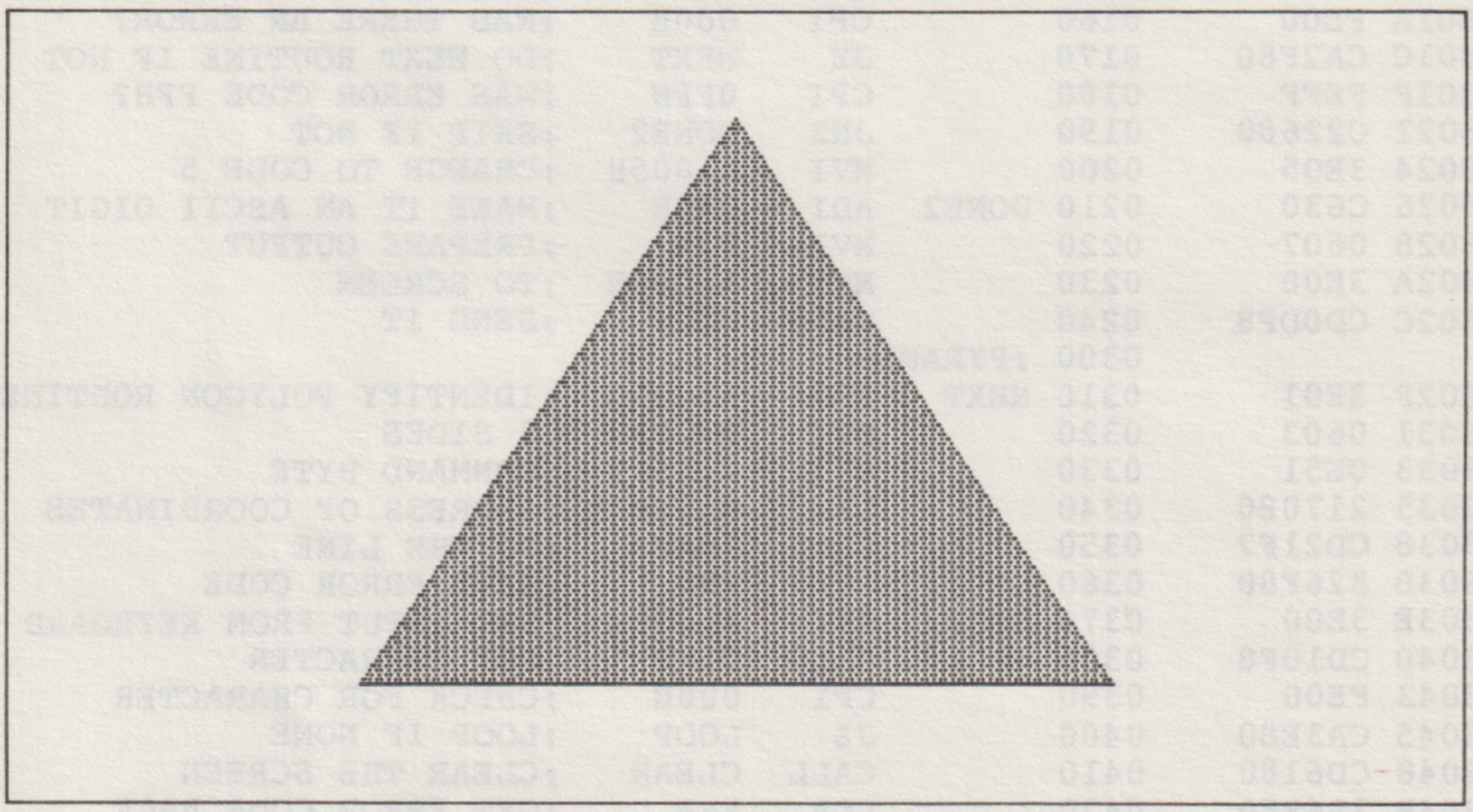
TRIANGLE - Sample Program

The following program draws a triangle on the ADVANTAGE video screen. The POLYGON routine is used to draw the triangle.

```
                0010          ;SET VIEWPORT
8000            0020          ORG   08000H
F721 =          0030 GRAF     EQU   0F721H  ;GRAPHICS ENTRY POINT
F80D =          0040 COUT     EQU   0F80DH  ;OUTPUT ENTRY POINT
F810 =          0050 CINP     EQU   0F810H  ;INPUT ENTRY POINT
8000 3E02       0060 SETVP    MVI   A,002H  ;IDENTIFY RECTANGLE
8002 0E80       0070          MVI   C,080H  ;VIEWPORT SETUP CODE
8004 216780     0080          LXI   H,VP    ;VIEWPORT COORDINATES
8007 CD21F7     0090 DOIT     CALL  GRAF    ;CALL GRAPHICS MANAGER
800A 326F80     0100          STA   ERR     ;SAVE ERROR CODE
800D 3E00       0110 WAIT     MVI   A,000H  ;SPECIFY KEYBOARD
800F CD10F8     0120          CALL  CINP    ;INPUT CHARACTER
8012 FE00       0130          CPI   000H    ;CHECK FOR INPUT
8014 CA0D80     0140          JZ    WAIT    ;LOOP IF NONE
8017 3A6F80     0150 DONE     LDA   ERR     ;GET ERROR CODE
801A FE00       0160          CPI   000H    ;WAS THERE AN ERROR?
801C CA2F80     0170          JZ    NEXT    ;DO NEXT ROUTINE IF NOT
801F FEFF       0180          CPI   0FFH    ;WAS ERROR CODE FFH?
8021 C22680     0190          JNZ   DONE2   ;SKIP IF NOT
8024 3E05       0200          MVI   A,005H  ;CHANGE TO CODE 5
8026 C630       0210 DONE2    ADI   030H    ;MAKE IT AN ASCII DIGIT
8028 0607       0220          MVI   B,A     ;PREPARE OUTPUT
802A 3E00       0230          MVI   A,000H  ;TO SCREEN
802C CD0DF8     0240          CALL  COUT    ;SEND IT
                0300 ;TRIANGLE
802F 3E01       0310 NEXT     MVI   A,001H  ;IDENTIFY POLYGON ROUTINE
8031 0603       0320          MVI   B,003H  ;3 SIDES
8033 0E51       0330          MVI   C,051H  ;COMMAND BYTE
8035 217080     0340          LXI   H,TRIAN ;ADDRESS OF COORDINATES
8038 CD21F7     0350          CALL  GRAF    ;DO THE LINE
803B 326F80     0360          STA   ERR     ;SAVE ERROR CODE
803E 3E00       0370 LOOP     MVI   A,000H  ;SET INPUT FROM KEYBOARD
8040 CD10F8     0380          CALL  CINP    ;GET CHARACTER
8043 FE00       0390          CPI   000H    ;CHECK FOR CHARACTER
8045 CA3E80     0400          JZ    LOOP    ;LOOP IF NONE
8048 CD6180     0410          CALL  CLEAR   ;CLEAR THE SCREEN
804B 3A6F80     0420          LDA   ERR     ;GET ERROR CODE BACK
804E FE00       0430          CPI   000H    ;WAS THERE AN ERROR?
8050 C8         0440          RZ            ;IF NOT, RETURN
8051 FEFF       0450          CPI   0FFH    ;WAS IT FFH?
8053 C25880     0460          JNZ   LOOP2   ;IF NO, SKIP
8056 3E05       0470          MVI   A,005H  ;CHANGE IT TO A 5
8058 C630       0480 LOOP2    ADI   030H    ;MAKE IT AN ASCII DIGIT
805A 47         0490          MOV   B,A     ;SET UP OUTPUT
```

```
805B 3E00      0500          MVI   A,000H  ;TO SCREEN
805D CD0DF8    0510          CALL  COUT    ;SEND IT
8060 C9        0520          RET           ;RETURN
8061 3E00      0700 CLEAR    MVI   A,000H  ;CLEAR SCREEN CODE
8063 CD21F7    0710          CALL  GRAF    ;SEND IT
8066 C9        0720          RET           ;GO BACK
8067 0000      6000 VP       DW    0000H   ;X1
8069 0000      6010          DW    0000H   ;Y1
806B 7F02      6020          DW    027FH   ;X2
806D F000      6030          DW    00EFH   ;Y2
806F 00        6040 ERR      DB    000H    ;LAST ERROR CODE
8070 3F01      7000 TRIAN    DW    013FH   ;X1
8072 3B00      7010          DW    003BH   ;Y1
8074 9F00      7020          DW    009FH   ;X2
8076 B300      7030          DW    00B3H   ;Y2
8078 DF01      7040          DW    01DFH   ;X3
807A B300      7050          DW    00B3H   ;Y3
```

PICTURE OF THE TRIANGLE DRAWN BY THE PRECEDING PROGRAM



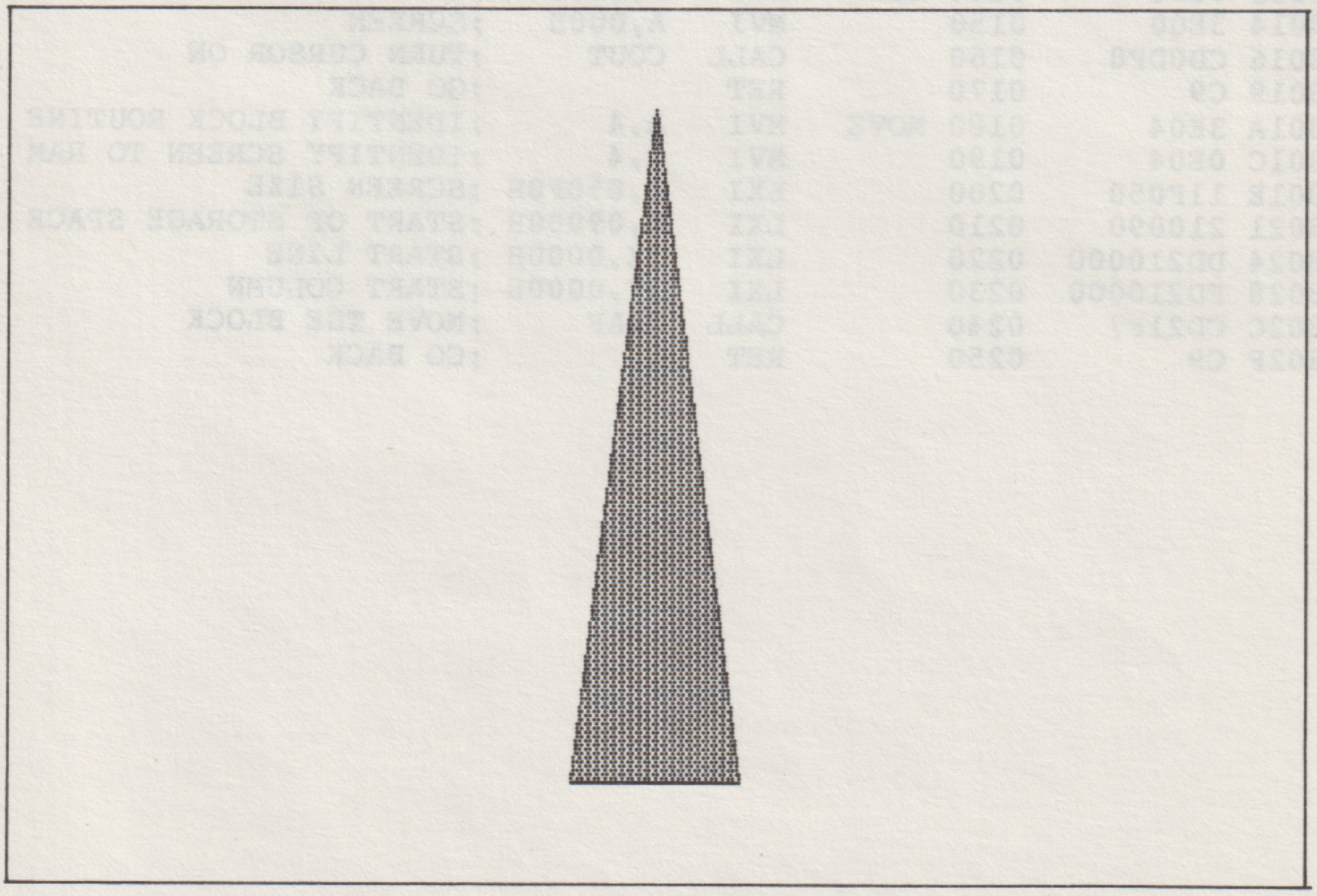
PYRAMID - Sample Program

The following program draws a pyramid on the ADVANTAGE video screen. The POLYGON routine is used to draw the pyramid.

```
                    0010          ;SET VIEWPORT
8000                0020          ORG   08000H
F721 =              0030 GRAF     EQU   0F721H  ;GRAPHICS ENTRY POINT
F80D =              0040 COUT     EQU   0F80DH  ;OUTPUT ENTRY POINT
F810 =              0050 CINP     EQU   0F810H  ;INPUT ENTRY POINT
8000 3E02           0060 SETVP    MVI   A,002H  ;IDENTIFY RECTANGLE
8002 0E80           0070          MVI   C,080H  ;VIEWPORT SETUP CODE
8004 216780         0080          LXI   H,VP    ;VIEWPORT COORDINATES
8007 CD21F7         0090 DOIT     CALL  GRAF    ;CALL GRAPHICS MANAGER
800A 326F80         0100          STA   ERR     ;SAVE ERROR CODE
800D 3E00           0110 WAIT     MVI   A,000H  ;SPECIFY KEYBOARD
800F CD10F8         0120          CALL  CINP    ;INPUT CHARACTER
8012 FE00           0130          CPI   000H    ;CHECK FOR INPUT
8014 CA0D80         0140          JZ    WAIT    ;LOOP IF NONE
8017 3A6F80         0150 DONE     LDA   ERR     ;GET ERROR CODE
801A FE00           0160          CPI   000H    ;WAS THERE AN ERROR?
801C CA2F80         0170          JZ    NEXT    ;DO NEXT ROUTINE IF NOT
801F FEFF           0180          CPI   0FFH    ;WAS ERROR CODE FFH?
8021 C22680         0190          JNZ   DONE2   ;SKIP IF NOT
8024 3E05           0200          MVI   A,005H  ;CHANGE TO CODE 5
8026 C630           0210 DONE2    ADI   030H    ;MAKE IT AN ASCII DIGIT
8028 0607           0220          MVI   B,A     ;PREPARE OUTPUT
802A 3E00           0230          MVI   A,000H  ;TO SCREEN
802C CD0DF8         0240          CALL  COUT    ;SEND IT
                    0300 ;PYRAMID
802F 3E01           0310 NEXT     MVI   A,001H  ;IDENTIFY POLYGON ROUTINE
8031 0603           0320          MVI   B,003H  ;3 SIDES
8033 0E51           0330          MVI   C,051H  ;COMMAND BYTE
8035 217080         0340          LXI   H,PYRAM ;ADDRESS OF COORDINATES
8038 CD21F7         0350          CALL  GRAF    ;DO THE LINE
803B 326F80         0360          STA   ERR     ;SAVE ERROR CODE
803E 3E00           0370 LOOP     MVI   A,000H  ;SET INPUT FROM KEYBOARD
8040 CD10F8         0380          CALL  CINP    ;GET CHARACTER
8043 FE00           0390          CPI   000H    ;CHECK FOR CHARACTER
8045 CA3E80         0400          JZ    LOOP    ;LOOP IF NONE
8048 CD6180         0410          CALL  CLEAR   ;CLEAR THE SCREEN
804B 3A6F80         0420          LDA   ERR     ;GET ERROR CODE BACK
804E FE00           0430          CPI   000H    ;WAS THERE AN ERROR?
8050 C8             0440          RZ            ;IF NOT, RETURN
8051 FEFF           0450          CPI   0FFH    ;WAS IT FFH?
8053 C25880         0460          JNZ   LOOP2   ;IF NO, SKIP
8056 3E05           0470          MVI   A,005H  ;CHANGE IT TO A 5
8058 C630           0480 LOOP2    ADI   030H    ;MAKE IT AN ASCII DIGIT
805A 47             0490          MOV   B,A     ;SET UP OUTPUT
```

```
805B 3E00      0500          MVI   A,000H  ;TO SCREEN
805D CD0DF8    0510          CALL  COUT    ;SEND IT
8060 C9        0520          RET           ;RETURN
8061 3E00      0700 CLEAR    MVI   A,000H  ;CLEAR SCREEN CODE
8063 CD21F7    0710          CALL  GRAF    ;SEND IT
8066 C9        0720          RET           ;GO BACK
8067 0000      6000 VP       DW    0000H   ;X1
8069 0000      6010          DW    0000H   ;Y1
806B 7F02      6020          DW    027FH   ;X2
806D F000      6030          DW    00EFH   ;Y2
806F 00        6040 ERR      DB    000H    ;LAST ERROR CODE
8070 1701      7000 PYRAM    DW    0117H   ;X1
8072 C800      7010          DW    00C8H   ;Y1
8074 6701      7020          DW    0167H   ;X2
8076 C800      7030          DW    00C8H   ;Y2
8078 3F01      7040          DW    013FH   ;X3
807A 2800      7050          DW    0028H   ;Y3
```

PICTURE OF THE PYRAMID DRAWN BY THE PRECEDING PROGRAM



BLOCK MOVE ROUTINE NUMBER 1 - SCREEN TO RAM STORAGE
Sample Program

The following assembly language program moves a copy of the ADVANTAGE video screen data to RAM storage at 9000H. No visible change in the video screen image occurs when this program is executed.

```
                0010          ;SET VIEWPORT
8000            0020          ORG   08000H
F721 =          0030 GRAF     EQU   0F721H   ;GRAPHICS ENTRY POINT
F80D =          0040 COUT     EQU   0F80DH   ;OUTPUT ENTRY POINT
F810 =          0050 CINP     EQU   0F810H   ;INPUT ENTRY POINT
8000 CD0A80     0060 BLOCK    CALL  COFF     ;TURN CURSOR OFF
8003 CD1A80     0070          CALL  MOVE     ;MOVE THE BLOCK
8006 CD1280     0080          CALL  CON      ;TURN CURSOR ON
8009 C9         0090          RET            ;GO BACK
800A 0619       0100 COFF     MVI   B,019H   ;CTRL-Y
800C 3E00       0110          MVI   A,000H   ;SCREEN
800E CD0DF8     0120          CALL  COUT     ;TURN CURSOR ON
8011 C9         0130          RET
8012 0618       0140 CON      MVI   B,018H   ;CTRL-X
8014 3E00       0150          MVI   A,000H   ;SCREEN
8016 CD0DF8     0160          CALL  COUT     ;TURN CURSOR ON
8019 C9         0170          RET            ;GO BACK
801A 3E04       0180 MOVE     MVI   A,4      ;IDENTIFY BLOCK ROUTINE
801C 0E04       0190          MVI   C,4      ;IDENTIFY SCREEN TO RAM
801E 11F050     0200          LXI   D,050F0H ;SCREEN SIZE
8021 210090     0210          LXI   H,09000H ;START OF STORAGE SPACE
8024 DD210000   0220          LXI   IX,0000H ;START LINE
8028 FD210000   0230          LXI   IY,0000H ;START COLUMN
802C CD21F7     0240          CALL  GRAF     ;MOVE THE BLOCK
802F C9         0250          RET            ;GO BACK
```

BLOCK MOVE ROUTINE NUMBER 2 - RAM STORAGE TO SCREEN
Sample Program

The following assembly language program will move the data stored at RAM address 9000H to the ADVANTAGE video screen. If the data stored at the specified RAM address consists of a previously saved video screen image, the image will appear on the ADVANTAGE video screen; if the data is not a previously saved video screen image, the results may be surprising.

```
                0010          ;MOVES RAM AT 9000-DFFFH ONTO SCREEM
8000            0020          ORG   08000H
F721 =          0030 GRAF     EQU   0F721H  ;ENTRY POINT FOR GRAPHICS
F80D =          0040 COUT     EQU   0F80DH  ;CHAR INPUT BYTE
F810 =          0050 CINP     EQU   0F810H  ;CHAR OUTPUT BYTE
8000 CD0A80     0060 BLOCK    CALL  COFF    ;TURN CURSOR OFF
8003 CD1A80     0070          CALL  MOVE    ;MOVE THE BLOCK
8006 CD1280     0080          CALL  CON     ;TURN CURSOR ON
8009 C9         0090          RET           ;GO BACK
800A 1E19       0100 COFF     MVI   E,019H  ;CTRL-Y
800C 0E00       0110          MVI   C,000H  ;SCREEN
800E CD0DF8     0120          CALL  COUT    ;TURN CURSOR ON
8011 C9         0130          RET
8012 1E18       0140 CON      MVI   E,018H  ;CTRL-X
8014 0E00       0150          MVI   C,000H  ;SCREEN
8016 CD0DF8     0160          CALL  COUT    ;TURN CURSOR ON
8019 C9         0170          RET           ;GO BACK
801A 3E04       0180 MOVE     MVI   A,4     ;IDENTIFY BLOCK ROUTINE
801C 0E00       0190          MVI   C,0     ;IDENTIFY RAM TO SCREEN
801E 11F050     0200          LXI   D,050F0H ;SCREEN SIZE
8021 210040     0210          LXI   H,04000H ;START OF STORAGE SPACE
8024 DD210000   0220          DB    0DDH,021H,000H,000H
8028 FD210000   0230          DB    0FDH,021H,000H,000H
802C CD21F7     0240          CALL  GRAF    ;MOVE THE BLOCK
802F C9         0250          RET           ;GO BACK
```

VALUES FOR COMMAND BYTE DISPLAY

PERIMETER ON	PERIMETER OFF
01=00000001	00=00000000
05=00000101	04=00000100
09=00001001	08=00001000
13=00001101	12=00001100
17=00010001	16=00010000
21=00010101	20=00010100
25=00011001	24=00011000
29=00011101	28=00011100
33=00100001	32=00100000
37=00100101	36=00100100
41=00101001	40=00101000
45=00101101	44=00101100
49=00110001	48=00110000
53=00110101	52=00110100
57=00111001	56=00111000
61=00111101	60=00111100

2.9 PERSONALIZING GDOS

The primary type of personalization that can be done to the GDOS is the insertion of input/output routines that allow communication of the GDOS and other system software with a particular hardware configuration. Input/output routine personalization is described in detail in the CHARACTER I/O SUB-SYSTEM section of this manual.

2.9.1 Additional GDOS Personalization

There are a number of other types of personalization that can be done to the GDOS that are described in this section.

READ AFTER WRITE CHECK

If the read after write check option is turned on, then a read and verify operation is performed after every disk write operation which checks that the data written on the disk by the write operation matches what is in RAM. With this option turned on, write operations will be slower, but read operations will be the same speed. It is strongly recomended that the read after write option be left on. The read after write option is turned on if the bit 0 of the byte stored at address F82BH of GDOS is 1 and turned off if zero. GDOS is supplied with this feature enabled.

PAGE SIZE

The output of some devices, such as CRT's and video displays, can only display a fixed size page of information at one time. If the page size option is enabled, then the file directory listing which is output by the LI command or the LIST library routine will stop after a page of information has been output and will not display the next page until the user indicates he wishes to proceed by typing the return key. The page size option only affects the operation of the screen (device #0). If the byte value at address F833H of the standard version of GDOS is zero, then the page size option is turned off and output will be continuous without stopping. If the value is non-zero, then the value is the number of lines on a page and the output will stop after that many less one lines of output have been displayed. The last line on the page will request the user to type return to continue.

Initially, the page size option is on and set for a page size of 24 lines.

AUTOMATIC START

If the automatic start option is turned on, then a single command which is stored in the GDOS input buffer is automatically executed immediately after a GDOS bootstrap operation. This feature, for example, allows for the automatic loading and running of a program such as BASIC upon system start-up. Initially, the automatic start option is turned off. To turn on the option, the byte value at address F830H of the standard version of GDOS should be set to zero. Initially, the value is one. Additionally, the input buffer must be setup to contain the command which should be automatically executed. The two addresses F831H and F832H contain the low order and high order byte, respectively, of the address of the input buffer within the GDOS. The input buffer should be loaded with the ASCII values of the successive characters of the desired command. The last character of the loaded command must be a return (0DH.)

EXAMPLE PERSONALIZATION

The following listing shows an example procedure which will modify the version of GDOS on a diskette so that the read after write option is turned on, the page size option is turned on and the page size is set to 24 lines, and the automatic start option is turned on and the automatic start command is set to be "GO GBASIC".

Load the diskette to be modified in drive #1

```
#LF GDOS 3800       Load GDOS into RAM
#GO M0000           Load and run the Monitor
>FM 382B 1          Turn on read after write option
>FM 3833 24T        Set page size to 24 lines
>FM 3830 0          Turn on automatic start option
>DH 3831,2          Determine input buffer address
3831: 50 FE         Address is FE50, for example
>FM 3E50 "G"        Load input buffer with "GO GBASIC"
>FM 3E51 "O"
>FM 3E52 " "
>FM 3E53 "G"
>FM 3E54 "B"
>FM 3E55 "A"
>FM 3E56 "S"
>FM 3E57 "I"
>FM 3E58 "C"
>FM 3E59 D          Put return code at end of command
>OS                 Return to GDOS
#SF GDOS 3800       Save modified GDOS back on the
                    diskette
```

2.9.2 GDOS Entry Points and Flags

```
                        GDOS ENTRY POINTS AND FLAGS

0000                ; THIS JUMP TABLE STARTS AT THE ADDRESS SHOWN IN
                    ; THE SYSTEMDISPATCH TABLE AS THE DISK
                    ; CONTROLLER ADDRESS
0000                ;
0000                ;
0000                        ORG     COLD
F700  C30000    U           JMP     COLD0 ; COLD BOOT
F703                ;
F703                        ORG     RAMSZ
F703  00                    DB      0 ; SIZE OF RAM FOUND BY BOOT IN
                                      UNITS OF 4 KBYTES
F709                ;
F709                ; THE FOLLOWING ROUTINE DEFINES THE CONTENTS
F709                ; OF ONE MAP REGISTER DURING INTERRUPT SERVICE.
F709                ; INPUT: B - REGISTER CONTENTS
F709                ;  C - REGISTER CODE NUMBER (SEE TABLE BELOW)
F709                ; AFFECTS PSW ONLY
F709                        ORG     IMDEF
F709  C30000    U           JMP     IMDF0
F70C                ;
F70C                ; THIS ROUTINE SETS UP AND PERFORMS AN OFTEN
                      CALL
F70C                ; WITH RAM MAPPING PRESCRIBED THROUGH IMDEF,
F70C                ; WITH THE STACK IN RAM THAT IS NOT MAPPED OUT,
F70C                ; AND WITH INTERRUPTS DISABLED
F70C                ; ALL REGISTERS ARE PRESERVED
F70C                        ORG     OFTNM
F70C  C30000    U           JMP     OFTM0
F70F                ;
F70F                ; THIS ROUTINE CAUSES A TRANSITION OF THE
                      SPEAKER CONE.
F70F                ; AFFECTS PSW ONLY
F70F                        ORG     SPTOG
F70F  C30000    U           JMP     SPTG0
F712                ;
F712                ; THIS ROUTINE WILL NOT START THE DRIVE MOTORS,
                      BUT
F712                ; IT WILL KEEP THEM FROM STOPPING FOR ANOTHER 3
                      SECONDS.
F712                ; ALL REGISTERS RESTORED
F712                        ORG     TIKLE
F712  C30000    U           JMP     TIKL0
F715                ;
F715                ; THE FOLLOWING TWO ROUTINES SHOULD ALWAYS BE
                      USED
F715                ; FOR READING OR WRITING ANY OF THE FOLLOWING
```

```
                        MACHINE
F715                  ; REGISTERS (OTHER THAN TO USE THE SPEAKER):
F715                  ; CODE          REGISTER
F715                  ; 0             MAP REGISTER 0 (00000H TO 03FFFH)
F715                  ; 1             MAP REGISTER 1 (04000H TO 07FFFH)
F715                  ; 2             MAP REGISTER 2 (08000H TO 0BFFFH)
F715                  ; 4             CONTROL REGISTER
F715                  ; 5             TOP DISPLAY SCAN LINE REGISTER
F715                  ;
F715                  ; INPUT: C - REGISTER CODE (FROM TABLE ABOVE)
F715                  ; OUTPUT: B - CURRENT CONTENTS
F715                  ; AFFECTS PSW AND B
F715                        ORG   RDREG
F715  C30000    U           JMP   RDRG0 ; READ A MACHINE REGISTER
F718                  ;
F718                  ; INPUT: C - REGISTER CODE (FROM TABLE ABOVE)
F718                  ;        B - NEW CONTENTS
F718                  ; AFFECTS PSW AND BC
F718                        ORG   WRREG
F718  C30000    U           JMP   WRRG0 ; WRITE A MACHINE REGISTER
F71B                  ;
F71B                  ; THE FOLLOWING TWO ROUTINES HAVE THE SAME
                        CALLING
F71B                  ; CONVENTIONS AS CHI AND CHO ACCEPT THAT THE
                        CONTENTS
F71B                  ; OF THE ACCUMULATOR ARE IGNORED.
F71B                        ORG   CHI0
F71B  C30000    U           JMP   CIN0 ; CONSOLE INPUT ROUTINE
F71E                  ;
F71E                        ORG   CHO0
F71E  C30000    U           JMP   COUT1 ; CONSOLE OUTPUT ROUTINE
F721                  ;
F721                  ; SEE GRAPHICS SUB-SYSTEM SECTION FOR
F721                  ; CALLING CONVENTIONS
F721                        ORG   GRAPH
F721  C30000    U           JMP   GMGR ; GRAPHICS DRIVER ENTRY POINT
F724                  ;
F724                  ;
F724                  ; THIS TEN BYTE BLOCK DEFINES THE
F724                  ; PIXELS THAT MAKE UP THE CURSOR
F724                        ORG   CTEMP
F724  FFFFFFFF              DW    0FFFFH,0FFFFH,0FFFFH,0FFFFH,0FFFFH
      FFFFFFFF
      FFFF
F72E                  ;
F72E                  ;
F72E                  ;
```

```
F72E                    ;
F72E                    ; SYSTEM DISPATCH TABLE
F72E                    ;
F72E                    ; THE ORIGIN OF THIS TABLE MUST ALWAYS
F72E                    ; BE A MULTIPLE OF 100H
F72E                    ;
F72E                          ORG    DSPCH
F800                    ;
F800                    BASE   EQU    $
F800                    ;
F800                    ; THE FIRST FOUR BYTES ARE USED BY THE GDOS,
F800                    ; IF PRESENT, TO STORE THE CURRENT TRACKS
F800  F8                      DB     DSPCH/256 ; DIRECT BOOT TO PROPER
                                     ADDRESS
F801                    ;
F801                    ; THIS SEQUENTIAL REVISION NUMBER CHANGES
F801                    ; WITH EACH NEW RELEASE OF THE SOFTWARE
F801                          ORG    REVN
F804  00          U           DB     RLEAS
F805                    ;
F805                    ; THE FOLLOWING BYTE IS RESERVED FOR FUTURE USE
F805                    ;RESV1: EQU $
F805  00                      DB     0
F806                    ;
F806                    ; GDOS STORES CURRENTLY SELECTED
F806                    ; DRIVE NUMBER HERE
F806                          ORG    SUNIT
F806  01                      DB     1 ; NORMAL BOOT DRIVE
F807                    ;
F807                    ; THE OFTEN ROUTINE IS CALLED FREQUENTLY DURING
F807                    ; USE OF THE DISK SYSTEMS
F807                    ; ONLY ACC AND FLAGS MAY BE MODIFIED
F807                    ; 12 BYTES OF STACK CAN BE USED
F807                    ; STACK MUST BE IN CURRENTLY ACTIVE RAM
F807                    ; MUST NOT BRANCH ANYWHERE DURING COLD BOOT
F807                          ORG    OFTEN
F807  C9                      RET    ; JUST RET DURING BOOT
F808  0000        U           DW     POLE ; JMP ADDR FOR POLLING
F80A                    ;
F80A                    ; THIS ENTRY POINT IS USED
F80A                    ; WHEN THE SYSTEM IS BOOTED DIRECTLY
F80A                    ; FROM A MICRO DISK
F80A                          ORG    CBOOT
F80A  C30000      U           JMP    START
F80D                    ;
F80D                    ; THIS IS THE CHARACTER OUTPUT ROUTINE
F80D                    ; IT IS CALLED WITH THE CHARACTER IN B AND
F80D                    ; WITH THE DEVICE # IN A
```

```
F80D                ; ONLY ACC AND FLAGS MAY BE MODIFIED
F80D                ; MUST RETURN THE SAME CHARACTER IN A
F80D                      ORG    CHO
F80D  C30000    U         JMP    DAOT ; ADDITIONAL DEVICE ROUTING
                                        ROUTINE
F810                ;
F810                ; THIS IS THE CHARACTER INPUT ROUTINE
F810                ; IT IS CALLED WITH THE DEVICE # IN A
F810                ; ONLY ACC AND FLAGS MAY BE MODIFIED
F810                ; MUST RETURN INPUT CHARACTER IN A
F810                      ORG    CHI
F810  C30000    U         JMP    CIN
F813                ;
F813                ; THIS NEXT ROUTINE IS CALLED ONCE AT INIT TIME
F813                ; IT CAN THEN USE ALL REGISTERS AND SHOULD
F813                ; PERFORM ANY NEEDED INITIALIZATION
F813                      ORG    INIT
F813  C30000    U         JMP    TINIT
F816                ;
F816                ; THIS IS THE CONTROL C ROUTINE
F816                ; EITHER THIS OR ISTAT IS CALLED FREQUENTLY
F816                ; DURING EXECUTION OF ANY NORMAL SOFTWARE
F816                ; ALL REGISTERS MAY BE USED
F816                ; IF NO INPUT DATA AT DEVICE 0 THEN
F816                ; RETURN BOTH Z AND C FLAGS FALSE
F816                ; IF DATA IS AVAILABLE IT IS RETURNED
F816                ; IN A WITH C FLAG TRUE
F816                ; RETURNS Z TRUE ONLY IF DATA IS CONTROL C
F816                      ORG    CON
F816  C30000    U         JMP    CONC
F819                ;
F819                ; MICRO DISK ERRORS JMP THRU THIS OR OTHER ERROR
                    JMP
F819                      ORG    HDERR
F819  C36680              JMP    08066H ; RETURN TO BOOT PROM ON
                                          ERROR DURING BOOT
F81C                ;
F81C                ; THIS IS THE MICRO DISK FILE LOOKUP ROUTINE
F81C                ; A MUST CONTAIN DEFAULT DRIVE(NORMALLY 1)
F81C                ; HL=POINTER TO FILE NAME IN RAM
F81C                ;    WITH OPTIONAL DRIVE NUMBER
F81C                ;    TERMINATED WITH BLANK OR CR
F81C                ; DRIVE NUMBER RETURNED IN A IF FILENAME
F81C                ;    SYNTAX OK, ELSE ZERO RETURNED
F81C                ; IF FOUND IN DIRECTORY THEN
F81C                ;    CARRY RETURNED FALSE AND
F81C                ;    HL=POINTER TO BYTE 8 OF ENTRY
F81C                ; IF NOT FOUND THEN
```

```
F81C                    ;  CARRY RETURNED TRUE AND
F81C                    ;  HL=FIRST FREE DISK ADDRESS
F81C                         ORG    DLOOK
F81C  C30000    U            JMP    DLOK
F81F                    ;
F81F                    ; THIS ROUTINE WRITES UPDATED DIRECTORY TO MICRO
                        DISK
F81F                    ; MUST FOLLOW DLOOK
F81F                         ORG    DWRIT
F81F  C30000    U            JMP    DWRI
F822                    ;
F822                    ; GENERAL MICRO DISK COMMAND ROUTINE
F822                    ; ACC= NUMBER OF SECTORS
F822                    ; B= COMMAND (0=WR, 1=RD, 2=VERIFY)
F822                    ; C= DRIVE, BIT 7=DOUBLE DENSITY (MUST BE TRUE)
F822                    ; DE= STARTING RAM ADDRESS
F822                    ; HL= STARTING DISK ADDRESS
F822                    ; RETURNS WITH CARRY TRUE IF BAD ARGS
F822                         ORG    DCOM
F822  C30000    U            JMP    DCOM0
F825                    ;
F825                    ; THIS ROUTINE LISTS MICRO DISK DIRECTORIES
F825                    ; ACC= DRIVE NUMBER
F825                    ; L= OUTPUT DEVICE NUMBER
F825                         ORG    DLIST
F825  C30000    U            JMP    LIST
F828                    ;
F828                    ; THIS IS THE RESTART ENTRY POINT
F828                         ORG    RSTRT
F828  C30000    U            JMP    RST0
F82B                    ;
F82B                    ; BIT 0 OF THIS FLAG CONTROLS THE
F82B                    ; READ AFTER WRITE CHECK OPTION ON
F82B                    ; MICRO DISKS ONLY
F82B                    ; READ AFTER WRITE IS ALWAYS DONE
F82B                    ; ON THE HARD DISK
F82B                    ; IF 1 THEN CHECK ON FLOPPIES ALSO
F82B                    ;
F82B                    ; BIT 7 OF THIS FLAG IS 1 ONLY IF
F82B                    ; INTERRUPTS SHOULD BE LEFT ENABLED
F82B                    ; AFTER ANY CODE WHICH MUST DISABLE THEM
F82B                         ORG    RWCHK
F82B  81                     DB     81H
F82C                    ;
F82C                    ; MICRO DISK ERRORS JMP THRU THIS OR OTHER ERROR
                        JMP
F82C                         ORG    DOSER
F82C  C30000    U            JMP    ERROR
```

```
F82F                    ;
F82F                    ; THIS BYTE SET TO DENSITY OF DIRECTORY
F82F                    ; BY DLOOK CALLS
F82F                    ; 0 IF SINGLE ; 80H IF DOUBLE
F82F                          ORG     DEN
F82F  80                      DB      80H ; ALWAYS  DOUBLE  IN  THIS
                                            COMPUTER
F830                    ;
F830                    ; THIS FLAG BYTE CONTROLS THE AUTOSTART FEATURE
F830                    ; OF THE COMMAND PROCESSOR.  THIS BYTE IS TESTED
F830                    ; AND SET TO ONE WHENEVER THE COMMAND PROCESSOR
F830                    ; IS EXECUTED.  IF THIS BYTE WAS ZERO THE
                          COMMAND
F830                    ; PROCESSOR WILL AUTOMATICALLY EXECUTE THE COMMAND
F830                    ; IN ITS INPUT BUFFER.  THIS FEATURE SHOULD BE
F830                    ; USED FOR TURNKEY STARTUP OF ANY SYSTEM.
F830                          ORG     AUTOS
F830  01                      DB      1
F831                    ;
F831                    ; THIS WORD POINTS TO THE TEXT LINE BUFFER USED
                          BY
F831                    ; THE COMMAND PROCESSOR.  THIS DATA IS PROVIDED
                          FOR
F831                    ; USE BY THE PERSON WHO PERSONALIZES A BOOTSTRAP
F831                    ; DISKETTE FOR TURNKEY STARTUP.
F831  0000          U         DW      CLINE
F833                    ;
F833                    ; THIS BYTE IS SCREEN LENGTH OF CONSOLE
F833                    ; USE ZERO IF HARD COPY TERMINAL
F833                          ORG     PAGES
F833  18                      DB      24
F834                    ;
F834                    ; THIS BYTE SHOWS MICRO DISK DRIVE COMBINATION.
F834                    ; THIS BYTE IS SET BY COLD BOOT ACCORDING TO
                          DRIVE ONE
F834                    ; UNLESS IT HAS AN INITIAL VALUE OTHER THAN
                          0FFH.
F834                    ; IF DISSIMILAR DRIVES ARE TO BE USED, THIS BYTE
                          MUST BE
F834                    ; CORRECTED MANUALLY ON THE BOOT DISKETTE.
F834                    ; SEE INSTRUCTIONS FOR FORMAT
F834                          ORG     CONFG
F834  FF                      DB      0FFH
F835                    ;
F835                    ; THE RESULT CODE OF EACH USE OF THE FILE
                          MANAGER
F835                    ; OTHER THAN FMABT IS STORED HERE FOR USE BY THE
                          COMMAND
```

```
F835                    ; PROCESSOR OR OTHER SOFTWARE WHICH REPORTS
                          ERRORS
F835                          ORG     RESLT
F835  00                      DB      MOK
F836                    ;
F836                    ; THIS BYTE CONTAINS THE LAST ERROR CODE NUMBER
F836                    ; RETURNED TO THE FILE MANAGER BY HDCOM
F836                    ; THE COMMAND PROCESSOR ZEROS THIS BYTE
F836                    ; WHEN THE ERROR IS REPORTED
F836                          ORG     HDEMC
F836  00                      DB      0
F837                    ;
F837                    ; THIS WORD CONTAINS THE ADDRESS OF THE LAST
F837                    ; SECTOR ACCESS ATTEMPTED BY THE FILE MANAGER
F837                          ORG     HDEDA
F837  0000                    DW      0
F839                    ;
F839                    ; THIS THE NUMBER OF THE LAST HARD DISK
F839                    ; DRIVE ACCESSED BY THE FILE MANAGER
F839                          ORG     HDEDN
F839  00                      DB      0
F83A                    ;
F83A                    ; THIS BYTE SHOWS THE MOST SIGNIFICANT
F83A                    ; BYTE OF BOOT ADDRESS AND OF THIS
F83A                    ; JUMP TABLE
F83A                          ORG     MDCB
F83A  F7                      DB      BADDR/256
F83B                    ;
F83B                    ; THE FOLLOWING BYTE IS RESERVED FOR FUTURE USE
F83B                    ;RESV2: EQU $
F83B  00                      DB      0
F83C                    ;
F83C                    ; THIS BYTE CONTAINS THE ADDRESS OF THE FIRST
F83C                    ; PAGE OF MEMORY WHICH SHOULD BE CONSIDERED
F83C                    ; BY USER SOFTWARE TO BE BEYOND THE UPPER LIMIT
F83C                          ORG     HMEM
F83C  E0                      DB      MTOP/256
F83D                    ;
F83D                    ; THIS  BYTE  CONTAINS  THE  ADDITIONAL  OUTPUT
                          DEVICE NUMBER.
F83D                    ; WHEN THIS BYTE IS NONZERO, ALL OUTPUT TO THE
                          MAIN CONSOLE
F83D                    ; (DEVICE ZERO) WILL BE ECHOED TO THE DEVICE
                          SPECIFIED HERE.
F83D                    ; THIS BYTE IS SET BY THE OD COMMAND.
F83D                          ORG     ADEV
F83D  00                      DB      0
F83E                    ;
```

```
F83E                ; TO ENABLE THE ADDITIONAL OUTPUT DEVICE
                      FEATURE, THE JUMP
F83E                ; TO THE ACTUAL CHARACTER OUTPUT ROUTINE IS
                      PLACED HERE, INSTEAD
F83E                ; OF AT CHO, ABOVE.
F83E                      ORG    AOUT
F83E  C30000    U         JMP    COUT
F841                ;
F841                ; THIS IS THE INPUT STATUS ROUTINE
F841                ; IT IS CALLED WITH THE DEVICE # IN A
F841                ; RETURNS NUMBER OF DEVICE TESTED IN A
F841                ; RETURNS Z FLAG TRUE IF INPUT DATA AVAILABLE
F841                ; NO OTHER REGISTERS MAY BE USED
F841                      ORG    ISTAT
F841  C30000    U         JMP    IST
F844                ;
F844                ; THIS IS THE OUTPUT STATUS ROUTINE
F844                ; IT IS CALLED WITH THE DEVICE # IN A
F844                ; RETURNS NUMBER OF DEVICE TESTED IN A
F844                ; RETURNS Z FLAG TRUE IF OUTPUT DEVICE READY
F844                ; NO OTHER REGISTERS MAY BE MODIFIED
F844                ; ISTAT AND OSTAT MAY BE USED BY SOFTWARE TO
F844                ; DETERMINE WHICH DEVICE NUMBERS ARE IMPLEMENTED
F844                      ORG    OSTAT
F844  C30000    U         JMP    OST
F847                ;
F847                ;
F847                ; THIS IS THE NORTH STAR LINE EDITOR
F847                ; ON ENTRY:
F847                ;    A= I/O DEVICE NUMBER
F847                ;    C= LENGTH OF INPUT BUFFER
F847                ;    DE= ADDR OF OLD LINE
F847                ;        TERMINATED WITH CR
F847                ;    HL= ADDR OF INPUT BUFFER
F847                ; ON EXIT:
F847                ;    HL, DE, AND B RESTORED
F847                ;    C= SPACE UNUSED IN INPUT BUFFER
F847                ;    A= RESULT CODE:
F847                ;       0: RETURN ENTERED
F847                ;       1: CONTROL C ENTERED
F847                ;       2: @ OR CONTROL N ENTERED
F847                ;       3: TOO MANY CHARS ENTERED
F847                ;    OLD LINE IS NOT CHANGED
F847                ;    CRLF IS NOT ECHOED
F847                ;    NEW LINE ENDS WITH A CR
F847                      ORG    LINED
F847  C30000    U         JMP    LNEDT
F84A                ;
```

```
F84A                    ; THIS IS THE ENTRY POINT TO THE
F84A                    ; HARD DISK FILE MANAGER
F84A                            ORG     FMNGR
F84A  AF                        XRA     A
F84B  3D                        DCR     A ; SHOW ERROR
F84C  C9                        RET
F84D                    ;
F84D                    ; THIS IS THE LOW LEVEL HARD DISK
F84D                    ; ACCESS ROUTINE
F84D                    ; THIS ROUTINE SHOULD NOT BE USED
F84D                    ; BY ANY NORMAL SOFTWARE
F84D                            ORG     HDCOM
F84D  067D                      MVI     B,MCNPR
F84F  C9                        RET
F850                    ;
F850                    ;
F850                    ; NOTHING BEYOND THIS POINT SHOULD BE CONSIDERED
F850                    ; FIXED AND INDEPENDENT OF REVISION LEVEL.
F850                    ; ****************************************
F850                    ;
```

2.9.3 Personalizing a New Diskette From an Old Diskette

GDOS software diskettes, as distributed, contain input/output routines which are tailored to the ADVANTAGE. Unless an additional peripheral which requires special input/output treatment is to be used with the ADVANTAGE, there is no need to personalize these routines. However, if it is desired to personalize a new system software diskette using the same personalized input/output routines that already exist on an old diskette, then the following procedure can be used to incorporate the old routines into the new software. The following listing gives the GDOS and Monitor commands which should be exactly followed to successfully copy the input/output routines.

In order to understand the personalization process, it is necessary to understand that the GDOS is normally loaded into RAM by the bootstrap PROM in two stages. The first 2K (800 hex) bytes of the GDOS file are loaded into RAM from F800H to FFFFH. Control is then passed to this portion of memory, and this part of the program loads the remainder of the GDOS into RAM from E000H to F7FFH. Thus, the system dispatch table, which will load at F800H, is found at the beginning of the GDOS file, and the computer dependent jump table, which will load at F700H is found near the end of the GDOS file. Because of this arrangement, loading the GDOS file into RAM at an odd multiple of 800H makes addresses within it easier to find for examination and modification.

Using an old diskette, do a system start-up sequence, then:

```
#LF GDOS 3800                Load copy of old GDOS
```

Next, remove the old diskette and insert a copy of the new diskette in drive #1.

```
#LF GDOS 1800                Load copy of new GDOS
#GO M0000                    Run the system monitor
>MM 5600,100 3600           Move I/O routines from old to
                             new GDOS
>MM 380D,C 180D             Move jumps from old to new GDOS
>MM 383E,9 183E
```

If the old GDOS has additional personalization, copy it now.

```
>OS                          Return to GDOS
#SF GDOS 1800                Write new GDOS onto new
                             diskette
#IL                          Restart the ADVANTAGE using the
                             newly created copy of GDOS.
```

2.9.4 ADVANTAGE Input/Output Routines

```
                    ADVANTAGE INPUT/OUTPUT ROUTINES

I/O BLOCK ************************************************************
F600                ; MUST START ON A 100H BOUNDARY
F600                ;
F600                ; IOBLK: EQU $
F600                ;
F600                ;
00F7                BLIDS   EQU     0F7H ; SIO BOARDLET ID BYTE
00DF                BLIDP   EQU     0DFH ; PIO BOARDLET ID BYTE
F600                ;
F600                ;
F600                ; PORT ADDRESS ROUTINE
F600                ; ENTER WITH DEVICE # IN ACC
F600                ; RETURNS BOARDLET ID BYTE IN ACC (0FFH IF NONE)
F600                ; IF BOARDLET THEN ALSO RETURNS BASE PORT
                                                        ADDRESS IN C
F600                ; RETURNS Z TRUE FOR SIO BOARDLET ONLY
F600                ; AFFECTS PSW AND BC
F600                ;
F600                BLAD0   EQU     $       ; REENTRY POINT SUBSTITUTES
                                                        DEVICE ZERO
F600  AF                    XRA     A
F601                ;
F601                BLADR   EQU     $       ; STANDARD ENTRY POINT
F601  D601                  SUI     1               ; GET 0FFH WITH CARRY
                                                    IF DEVICE ZERO
F603  D8                    RC
F604  FE06                  CPI     6               ; TEST FOR DEVICE 1
                                                        THROUGH 6
F606  30F8                  JRNC    BLAD0       ; ELSE SUBSTITUTE DEVICE
                                                        ZERO
F608  2F                    CMA     ; CONVERT TO -6 THROUGH -1
F609  C676                  ADI     76H             ; GENERATE ID PORT
                                                        ADDRESS
F60B  4F                    MOV     C,A
F60C  ED40                  CIN     B               ; GET BOARDLET ID TO B
                                                    (0FFH IF NONE)
F60E  87                    ADD     A
F60F  87                    ADD     A           ; GENERATE BASE PORT
                                                        ADDRESS
F610  87                    ADD     A
F611  87                    ADD     A
F612  4F                    MOV     C,A
F613  78                    MOV     A,B
F614  FEF7                  CPI     BLIDS       ; TEST FOR SIO BOARDLET
F616  C9                    RET
F617                ;
```

```
F617                ;
F617                ; INPUT STATUS CHECK ROUTINE
F617                ;
F617                IST     EQU     $
F617  CD0CF7                CALL    OFTNM       ; TAKE CARE OF BUSINESS
F61A                ;
F61A                ISTA    EQU     $           ; ALTERNATIVE ENTRY POINT
F61A  C5                    PUSH    B
F61B  F5                    PUSH    PSW
F61C  CD01F6                CALL    BLADR       ; CONVERT DEVICE TO PORT
                                                      ADDRESS AND ID
F61F  2812                  JRZ     ISTS         ; JUMP IF SERIAL
                                                           BOARDLET
F621  FEDF                  CPI     BLIDP        ; TEST FOR PARALLEL
                                                           BOARDLET
F623  280A                  JRZ     ISTP
F625                ;
F625                IST0    EQU     $           ; KEYBOARD STATUS CHECK
F625  DBD0                  IN      SPRCS           ; READ OTHER PROCESSOR
                                                           STATUS
F627  2F                    CMA     ; MAKE KEYBOARD STATUS LOW TRUE
F628                ;
F628                ST0X    EQU     $           ; DEVICE ZERO STATUS CHECK
                                                           EXIT
F628  E640                  ANI     CHRDY        ; SET OR RESET Z FLAG
F62A  3E00                  MVI     A,0          ; SHOW DEVICE ZERO
                                                           TESTED
F62C  C1                    POP     B            ; FORMER PSW
F62D  C1                    POP     B
F62E  C9                    RET
F62F                ;
F62F                ISTP    EQU     $           ; PIO INPUT STATUS CHECK
F62F  0608                  MVI     B,8          ; PI FLAG BIT MASK
F631  1802                  JR      BSTP
F633                ;
F633                ISTS    EQU     $           ; SIO INPUT STATUS CHECK
F633  0602                  MVI     B,2          ; RDA BIT MASK
F635                ;
F635                BSTS    EQU     $           ; SIO STATUS CHECK
F635                BSTP    EQU     $           ; PIO STATUS CHECK
F635  0C                    INR     C               ; ADVANCE TO STATUS
                                                       PORT ADDRESS
F636                ;
F636                BSTCK   EQU     $           ; BOARDLET STATUS CHECK
F636  ED78                  CIN     A               ; READ STATUS BYTE
F638  2F                    CMA     ; MAKE ALL STATUS BITS LOW TRUE
F639  A0                    ANA     B               ; TEST SPECIFIED STATUS
                                                           BIT
```

```
F63A  C1              POP   B          ; FORMER PSW
F63B  78              MOV   A,B        ; I/O DEVICE NUMBER
F63C  C1              POP   B
F63D  C9              RET
F63E                ;
F63E                ; OUTPUT STATUS CHECK ROUTINE
F63E                ;
F63E              OST EQU   $
F63E  C5              PUSH  B
F63F  F5              PUSH  PSW
F640  CD01F6          CALL  BLADR      ; CONVERT DEVICE NUMBER
                                         TO PORT ADDRESS
F643  280A            JRZ   OSTS       ; JUMP IF SERIAL
                                         BOARDLET
F645  FEDF            CPI   BLIDP      ; TEST FOR PARALLEL
                                         BOARDLET
F647  3E00            MVI   A,0
F649  20DD            JRNZ  ST0X       ; IF NEITHER, SHOW
                                         DEVICE ZERO READY
F64B                ;
F64B              OSTP EQU  $          ; PIO OUTPUT STATUS CHECK
F64B  0604            MVI   B,4        ; PO FLAG BIT MASK
F64D  18E6            JR    BSTP       ; EXIT THROUGH PIO
                                         STATUS CHECK
F64F                ;
F64F              OSTS EQU  $          ; SIO OUTPUT STATUS CHECK
F64F  0601            MVI   B,1        ; TBE BIT MASK
F651  18E2            JR    BSTS       ; EXIT THROUGH SIO
                                         STATUS CHECK
F653                ;
F653                ;
F653                ; CHARACTER OUTPUT ROUTINE
F653                ;
F653              ZCOUT EQU   $        ; ALTERNATIVE ENTRY POINT
F653  3E00            MVI   A,0        ; SUBSTITUTE FIXED
                                         DEVICE NUMBER
F655                ;
F655              COUT EQU    $        ; STANDARD ENTRY POINT
F655  CD3EF6          CALL  OST        ; CHECK STATUS OF
                                         SPECIFIED DEVICE
F658  20FB            JRNZ  COUT       ; LOOP UNTIL READY FOR
                                         DATA
F65A  B7              ORA   A          ; TEST DEVICE NUMBER
F65B  CA1EF7          JZ    CHO0       ; USE STANDARD ROUTINE
                                         FOR CONSOLE OUTPUT
F65E  C5              PUSH  B
F65F  C5              PUSH  B
F660  CD01F6          CALL  BLADR      ; GET PORT ADDRESS TO C
```

```
F663  D6E0            SUI   BLIDP+1     ; GET -1 ONLY IF PIO
F665  47              MOV   B,A         ; SAVE PIO FLAG
F666  F1              POP   PSW         ; GET CHARACTER TO SEND
                                                INTO ACC
F667  04              INR   B           ; TEST PIO FLAG
F668  CCA2F6          CZ    STRPR       ; PERFORM SPECIAL
                                              HANDLING FOR PIO
F66B  ED79            COUT  A         ; OUTPUT SIO OR PIO DATA
F66D  C1              POP   B
F66E  C9              RET
F66F                ;
F66F                ; CONTROL C CHECK
F66F                ;
F66F              CONC  EQU   $
F66F  3E00            MVI   A,0         ; MAIN CONSOLE DEVICE
                                                NUMBER
F671  CD17F6          CALL  IST         ; TEST STATUS OF CONSOLE
F674  37              STC
F675  3F              CMC     ; ENSURE CARRY FALSE
F676  C0              RNZ     ; RETURN FALSE IF NO CHARACTER
                                                TYPED
F677  CD88F6          CALL  CINN        ; INPUT THE CHARACTER
                                              THAT WAS FOUND
F67A  FE03            CPI   3           ; TEST FOR CONTROL C
F67C  37              STC     ; INDICATE THAT A CHARACTER WAS
                                                FOUND
F67D  C9              RET
F67E                ;
F67E                ;
F67E                ; CHARACTER INPUT ROUTINE
F67E                ;
F67E              ZCIN  EQU   $       ; ALTERNATIVE ENTRY POINT
F67E  3E00            MVI   A,0         ; SUBSTITUTE FIXED
                                              DEVICE NUMBER
F680                ;
F680              CIN   EQU   $         ; STANDARD ENTRY POINT
F680  CD12F7          CALL  TIKLE       ; KEEP DRIVES AWAKE IF
                                                USER IS
F683                ;
F683              CINL  EQU   $         ; INPUT STATUS LOOP
F683  CD17F6          CALL  IST         ; CHECK STATUS OF
                                              SPECIFIED DEVICE
F686  20FB            JRNZ  CINL        ; LOOP UNTIL DATA
                                              AVAILABLE
F688                ;
F688              CINN  EQU   $       ; INPUT IMMEDIATELY FROM
                                              DEVICE IN ACC
F688  B7              ORA   A         ; TEST DEVICE NUMBER
```

```
F689  CA1BF7              JZ      CHI0        ; USE STANDARD ROUTINE
                                                FOR CONSOLE INPUT
F68C  C5                  PUSH    B
F68D  CD01F6              CALL    BLADR     ; GET PORT ADDRESS TO C
F690  D6DF                SUI     BLIDP      ; TEST FOR PIO BOARDLET
F692  D601                SUI     1          ; SET CARRY IF PIO
                                                BOARDLET
F694  ED78                CIN     A          ; INPUT SIO OR PIO DATA
                                             (DOES NOT EFFECT CARRY)
F696  3006                JRNC    CINN1     ; SKIP AHEAD IF NOT PIO
F698  CBC9                STB     1,C        ; ADJUST TO PI FLAG
                                                RESET ADDRESS
F69A  CBD1                STB     2,C
F69C  ED79                COUT    A         ; RESET PI FLAG
F69E              CINN1   EQU     $
F69E  E67F                ANI     7FH        ; STRIP PARITY
F6A0  C1                  POP     B
F6A1  C9                  RET
F6A2              ;
F6A2              ;
F6A2              STRPR   EQU     $       ; PO FLAG RESET
F6A2  CBD1                STB     2,C       ; ADJUST TO PO FLAG
                                                RESET ADDRESS
F6A4  ED79                COUT    A          ; RESET PO FLAG
F6A6  CB91                RES     2,C        ; RESTORE PORT ADDRESS
F6A8              STRPO   EQU     $       ; HIGH BIT STROBE ROUTINE
                                                FOR PIO
F6A8  F680                ORI     80H        ; SET STROBE BIT FALSE
F6AA  ED79                COUT    A
F6AC  EE80                XRI     80H        ; TOGGLE STROBE
F6AE  ED79                COUT    A
F6B0  EE80                XRI     80H        ; TOGGLE STROBE
F6B2  C9                  RET     ; TO FINAL OUTPUT IN COMMON CODE
F6B3              ;
F6B3              ;
F6B3              ; USER'S INITIALIZATION ROUTINE
F6B3              ;
F6B3              TINIT   EQU     $
F6B3  21E1F6              LXI     H,TBLSI    ; POINT TO SIO
                                                INITIALIZATION TABLE
F6B6  AF                  XRA     A          ; SET UP LOOP COUNTER
F6B7  F5                  PUSH    PSW
F6B8              ;
F6B8              TINBL   EQU     $       ; LOOP THROUGH THE BOARDLET
                                                SLOTS
F6B8  F1                  POP     PSW        ; LOOP COUNTER
F6B9  3C                  INR     A          ; ADVANCE TO NEXT
                                                DEVICE
```

```
F6BA  FE07              CPI   7
F6BC  C8                RZ    ; EXIT AFTER DEVICE SIX
F6BD  F5                PUSH  PSW     ; LOOP COUNTER
F6BE  CD01F6            CALL  BLADR   ; GET BASE ADDRESS AND
                                        BOARDLET ID
F6C1  FEF7              CPI   BLIDS    ; TEST FOR SIO BOARDLET
F6C3  200B              JRNZ  TINPB    ; IF NOT, SKIP AHEAD TO
                                        PIO TEST
F6C5                ;
F6C5                TINSB EQU   $     ; SIO BOARDLET
                                        INITIALIZATION
F6C5  0C                INR   C          ; ADVANCE TO COMMAND
                                        ADDRESS
F6C6  0604              MVI   B,UINBC    ; BYTES REQUIRED TO
F6C8  EDB3              OUTIR ;      INITIALIZE THE USART
F6CA  CBD9              STB   3,C        ; ADJUST TO BAUD RATE
                                        ADDRESS
F6CC  EDA3              OUTI  ; SET BAUD RATE
F6CE  18E8              JR    TINBL       ; LOOP BACK FOR NEXT
                                        BOARDLET
F6D0                ;
F6D0                TINPB EQU   $     ; PIO BOARDLET
                                        INITIALIZATION
F6D0  110500            LXI   D,SINBC
F6D3  19                DAD   D          ; SKIP AN ENTRY IN SIO
                                        TABLE
F6D4  FEDF              CPI   BLIDP      ; TEST FOR PIO BOARDLET
F6D6  20E0              JRNZ  TINBL    ; IF NEITHER SIO NOR
                                        PIO, DON'T INITIALIZE
F6D8  3E0D              MVI   A,13       ; CARRIAGE RETURN TO
                                        START PARALLEL PRINTER
F6DA  CDA8F6            CALL  STRPO    ; SEND IT WITH HIGH BIT
                                        STROBE
F6DD  ED79              COUT  A          ; FINISH OUTPUT
F6DF  18D7              JR    TINBL      ; LOOP BACK FOR NEXT
                                        BOARDLET
F6E1                ;
F6E1                TBLSI EQU   $     ; SIO BOARDLET
                                        INITIALIZATION TABLE
F6E1  CE373737          DB    0CEH,037H,037H,037H   ; NORMAL,
                                        ASYNCHRONOUS
0004                UINBC EQU   $-TBLSI  ; USART INITIALIZATION
                                        BYTE COUNT
F6E5                ;
F6E5  70                DB    112                ; 1200 BAUD
                                        ON DEVICE ONE
0005                SINBC EQU   $-TBLSI  ; SIO INITIALIZATION BYTE
                                        COUNT
```

```
F6E6                    ;
F6E6  CE373737          DB      0CEH,037H,037H,037H   ; NORMAL,
                                                        ASYNCHRONOUS
F6EA  70                DB      112                   ; 1200 BAUD
                                                      ON DEVICE TWO
F6EB                    ;
F6EB  CE373737          DB      0CEH,037H,037H,037H   ; NORMAL,
                                                        ASYNCHRONOUS
F6EF  70                DB      112                   ; 1200 BAUD
                                                    ON DEVICE THREE
F6F0                    ;
F6F0  CE373737          DB      0CEH,037H,037H,037H   ; NORMAL,
                                                        ASYNCHRONOUS
F6F4  70                DB      112                   ; 1200 BAUD
                                                     ON DEVICE FOUR
F6F5                    ;
F6F5  CE373737          DB      0CEH,037H,037H,037H   ; NORMAL,
                                                        ASYNCHRONOUS
F6F9  70                DB      112                   ; 1200 BAUD
                                                     ON DEVICE FIVE
F6FA                    ;
F6FA  CE373737          DB      0CEH,037H,037H,037H   ; NORMAL,
                                                        ASYNCHRONOUS
F6FE  70                DB      112                   ; 1200 BAUD
                                                     ON DEVICE SIX
F6FF                    ;
F6FF                    ;
F6FF                    ;
F6FF                    ;
F6FF                    ;
F6FF                    ; ********************************* C2DSP *
```

3.1 INTRODUCTION

This chapter describes Version 2 of the North Star Monitor. The North Star Monitor is a program which provides the user with certain maintenance and debugging functions which would normally be provided in a limited way on systems which include a control panel.

Commands to the Monitor are entered via the console using a format consistent with the GDOS commands. The console is the ADVANTAGE video screen and keyboard. Console communication is done using the GDOS I/O routines. When the Monitor is in COMMAND MODE, i.e., is ready to accept a command, it will print a > at the beginning of a line on the console. Command editing facilities compatible with the North Star GBASIC editing features are included in the Monitor. The following list summarizes the commands available:

CM - Compare memory block contents
FM - Fill memory block
MM - Move memory block contents
SM - Search memory block
TM - Test memory block
DH - Display memory hexadecimal
DA - Display memory with ASCII interpretation
DS - Display memory and substitute values
JP - Jump to program
OS - Return control to the GDOS
IL - Perform initial load from bootstrap PROM
OD - Assign output device number for the Monitor
ID - Assign input device number for the Monitor

A detailed description of each command appears in a later section below. All printed output from the Monitor is formatted to fit into sixty-four character lines.

3.2 COMMAND FORMAT

No action is taken on any command until the command is fully entered and all editing functions are complete (i.e. when a carriage return is typed). Each command is fully checked for syntax errors before any part of the command is performed. Thus, a command aborted because of illegal syntax will have no effect.

The command syntax is quite simple. Each command appears on a single line and begins with a two letter mnemonic name which identifies the command. Upper case (capital) letters must be used for commands. The command name is followed by any required parameters which are separated from the command name and each other by spaces (blanks). Spaces may not appear within a parameter (except within quotation marks as described later). The command is terminated by a RETURN.

Commands may take no parameters or may take one or more of several types of parameters. The parameters required for Monitor commands have the following forms:

NUMBER — A NUMBER may be any of the following:

1. A hexadecimal number specified by a sequence of one or more hexadecimal digits.
2. A hexadecimal number specified by a sequence of one or more hexadecimal digits followed by the letter H.
3. A decimal number specified by a sequence of one or more decimal digits followed by the letter T.

ADDRESS — An ADDRESS is a NUMBER in the range from 0 though 65535T (0 through FFFFH).

BLOCK — A BLOCK is a contiguous region of memory specified by one of the following:

1. An ADDRESS alone specifies a one byte block at the given memory address.
2. Two ADDRESSES separated by a hyphen (-) specify a block whose first byte is at the first address given

and whose last byte is at the second address given (e.g., 2000-3FFF). The second address may not be less than the first.

3. An ADDRESS followed by a NUMBER separated by a comma specifies a block whose first byte is at the ADDRESS given and whose length is given by the NUMBER (e.g., 2000,1000). The address of the last byte (ADDRESS + NUMBER - 1) may not exceed 65535T (FFFFH).

BYTE VALUE

A BYTE VALUE is a parameter whose value fills a single byte. It may be specified by either of two forms:

1. A NUMBER in the range 0 though 255T (0 though FFH), or

2. A non-control character in quotation marks (e.g., "A"). A non-control character is any character entered without using the control key or other function keys such as carriage return, line feed, tab, etc.

DEVICE NUMBER

A DEVICE NUMBER is a cross-hatch (#), followed by a digit from 0 to 6, corresponding to one of the 7 possible I/O devices which have been interfaced to your GDOS through the GDOS I/O routines. Note that a DEVICE NUMBER may be given immediately after the code for any Monitor command which produces output. As a result, the output produced by the execution of that command will be printed on the specified device. For example, to send a hexadecimal dump of memory from 2000H-29FFH to system output device 4:

```
>DH #4 2000-29FF
```

Also note that only the output for the given command will be re-directed. After the command has been executed, output resumes on the regular Monitor output device (as set by the OD command).

3.3 MONITOR COMMANDS

This section gives a description and example for each Monitor command. An expanded example of the use of the Monitor follows in a later section. Note that the typing of commands can be corrected using all the line editor features described in the LINE EDITOR chapter of the GBASIC Manual.

3.3.1 CM - Compare Memory

COMMAND: CM <block> <address>

REMARKS: Compare the memory area defined by BLOCK with the area of the same length starting at ADDRESS and print the address and contents of all corresponding bytes which are not identical. For example, to compare the contents of the seven bytes starting at 3400H with the seven bytes starting at E385H:

```
>CM 3400,7 E385
```

3.3.2 FM - Fill Memory

COMMAND: CM <block> <byte value>

REMARKS: Fill each byte of the memory area defined by BLOCK with the value specified by the BYTE VALUE. For example, to fill the block of memory from 4000H to 5000H with the ASCII blank character:

```
>FM 4000-5000 " "
```

3.3.3 MM - Move Memory

COMMAND: MM <block> <address>

REMARKS: Move the contents of the memory area defined by BLOCK to the corresponding positions in the area of the same size beginning at ADDRESS. Overlapping areas of memory are allowed and the moves are performed correctly. For example, to move the contents of the one hundred (decimal) byte block starting at address 9000 (decimal) into the area starting at F000H:

```
>MM 9000T,100T F000
```

3.3.4 SM - Search Memory

COMMAND: SM <block> <list of byte values>

REMARKS: Search the area of memory defined by BLOCK and print the addresses of all occurrences of the sequence of bytes specified in the LIST OF BYTE VALUES. The list must contain at least one byte value. If it contains more than one, commas must separate the various byte values from one another. For example, to find all bytes in the standard GDOS area whose value is one:

```
>SM 2000-29FF 1
```

To find all occurrences of the sequence "MINE" in the block 4000H-5000H:

```
>SM 4000-5000 "M","I","N","E"
```

It is not necessary for the sequence of bytes defined in a LIST to be wholly contained within the specified BLOCK. SM will report the locations of all occurrences of the sequence as long as the first character in each occurrence lies within the BLOCK.

3.3.5 TM - Test Memory

COMMAND: TM <block> <number>

REMARKS: Test the memory area defined by BLOCK. The NUMBER parameter specifies the delay (in seconds) between a write phase of the test and the subsequent verify phase. The delay allows testing memory for failures resulting from gradual decay of data. The number may have a value from 0 to 255T (0 to FFH). For example, to test the 8K memory located at 6000H without delay between the write phase and the verify phase:

```
>TM 6000H,2000 0
```

The test operates as follows: Values are written into every byte in the block (write phase) and then the values are checked (verify phase). In a complete pass, 256 write and verify phases occur, so that every possible value is tested in every byte. The test is designed to catch addressing errors as well as data bit errors. The test

requires about 15 seconds per thousand bytes of memory for a complete pass (plus 256 times the delay value). The test will run continuously and can only be halted by typing a control-C. Note that even running the test for a few seconds will perform a quick check on the entire block. After each complete pass, the TM command will print PASS COMPLETE on the terminal. For every memory error detected, the address of the problem is printed in hexadecimal followed by the value stored at that address and the value found at that address.

3.3.6 DH - Display Hexadecimal

COMMAND: DH <block>

REMARKS: Display the contents of the memory area defined by BLOCK on the console in a format of two hexadecimal digits per byte with sixteen bytes on each line. For example, to print the contents of memory from 12000 (decimal) to 12099 (decimal) in hexadecimal:

```
>DH 12000T,100T
```

3.3.7 DA - Display ASCII

COMMAND: DA <block>

REMARKS: Display the contents of the memory area defined by BLOCK in the format of the DH command but with additional lines showing the ASCII character represented by the low order seven bits of each byte positioned immediately under the corresponding hexadecimal digits. A control code is printed as a blank and each character is preceded by a minus-sign if the high order bit of the byte is a one. For example, to print the contents of memory from 12000 (decimal) to 12099 (decimal) in both hexadecimal and as ASCII characters:

```
>DA 12000T-12099T
```

3.3.8 DS - Display and Substitute

COMMAND: DS <address>

REMARKS: Display the current contents of the memory area starting at ADDRESS in hexadecimal, one byte at a time, and allow the option of substituting a new value for each byte. After each byte is displayed,

if it is desired to modify the contents of the displayed cell, then a new hexadecimal value in the range 0 to FF may be entered. If it is desired to leave the byte unchanged, then type one of the following terminators immediately. Whether or not a new value is entered, the terminator character typed determines the subsequent action:

1) Blank. If a blank is typed then the next byte will be displayed for possible replacement.

2) Comma. If a comma is typed, then the printing of the next byte is suppressed, and subsequent typing will either replace or leave unmodified that next cell.

3) Carriage return. Typing a carriage return finishes the command and returns to command mode.

Note that the replacement occurs as soon as one of the terminators is typed, so that the line editor may not be used as in other commands. However, a typing error in specifying a new value may be corrected with any of the backspace commands before the terminator is typed. For example, to replace the contents (currently 64H) of the byte at 3233H with a zero:

```
>DS 3233
3233 64=0
```

3.3.9 JP - Jump

COMMAND: JP <address>

REMARKS: Cause the computer control to jump to the specified ADDRESS. The Monitor performs a CALL instruction to that address, so that if the program executed at that address later performs a RET and has preserved the stack pointer and not destroyed any of the memory of the Monitor, then control will return to the Monitor for another command. For example, to execute the normal continue location for GBASIC:

```
>JP 2A04
```

Note that when the JP command is executed, the HL register pair is set to point to the remainder of

the command which was typed in (any characters typed after the ADDRESS). This makes it possible to send parameters through the JP command to the program being executed.

3.3.10 OS - Exit to Operating System

COMMAND: OS

REMARKS: Jump to the standard re-entry address of the GDOS. This command requires no parameters.

3.3.11 IL - Initial Load

COMMAND: IL

REMARKS: Jump to the bootstrap startup PROM to perform an initial load of the GDOS from diskette. This command requires no parameters.

3.3.12 OD/ID - Set Output/Input Device

COMMAND: OD <device number>
ID <device number>

REMARKS: Sets the Monitor's output or input device number, respectively, to the specified DEVICE NUMBER. During Monitor output and input, these numbers are sent to the GDOS I/O routines, and may be used by them to determine routing of output and input to and from the appropriate system I/O devices. The value supplied to OD or ID must be in the range of 0 to 6. Both the input and output device numbers are set to zero when the Monitor is entered, corresponding to I/O from the console terminal. When an OD or ID command is executed, the new device number remains in effect for all output or input, respectively, produced by the Monitor until another OD or ID command is typed. (Note that these device numbers will also remain in effect upon re-entry to the Monitor at its initial entry point.) For example, to cause output to appear at output device 1:

```
>OD #1
```

To cause input to be accepted from device 6:

```
>ID #6
```

3.3.13 Interrupting the Monitor

Some commands may compute for long periods of time and/or produce substantial amounts of output. The CM (compare memory), SM (search memory), DH (display hexadecimal), DA (display ASCII) and TM (test memory) commands may be interrupted by typing a control-C on the console. This will interrupt the Monitor to allow a new command to be entered.

3.4 HARDWARE REQUIREMENTS

Any system operating the North Star GDOS and GBASIC will execute the Monitor. However, because of its applications, the Monitor is supplied in versions assembled at two different locations. Each version of the Monitor requires 2048 (800H) bytes and has its primary entry point at its origin. The standard versions are:

File name	Origin	Description
M0000	0000H	This version occupies memory locations 0000H-07FFH.
MC000	C000H	This version occupies memory locations C000H-C7FFH.

3.5 PERSONALIZING THE MONITOR

The Monitor begins with a series of jump instructions through which it is entered and through which it links to the GDOS input/output routines. Since the Monitor is supplied in two versions, note that the following discussion applies to the version with origin at zero.

Address	Instruction	Purpose
0000H	JMP INIT	This is the initial entry point to the Monitor
0003H	JMP F80DH	Jump to the GDOS COUT routine
0006H	JMP F810H	Jump to the GDOS CIN routine
0009H	JMP F828H	Jump to the GDOS re-entry point
000CH	JMP F816H	Jump to the GDOS CONTC routine
000FH	JMP F700H	Jump to the PROM bootstrap address

Consult the GETTING STARTED section for details of the required routines.

The value at address 0012H is the ASCII code for the character to be echoed when one of the following line editing control characters is typed: control-Q, underline, left-arrow, rubout, or delete. Initially, this value is set to the destructive backspace (7FH).

Provision has been made for adding as many as four user-defined commands to the Monitor. Approximately 240H bytes after the origin of the Monitor there is a block of seventeen (11H) bytes of zeros which lies at the end of the Monitor command table. This provides space for adding up to four entries of four bytes each to define user commands and allows for the zero byte which always terminates the command table.

Each command definition requires four bytes: the first pair of bytes contains the two-letter command name. The second pair of bytes contains the address (low-order byte first) of the routine to be called to execute the command. The program to implement the command should preserve the stack pointer and re-enter the Monitor with a RET instruction. No more than 30 bytes should be pushed onto the Monitor stack during execution of the command.

3.5.1 EXAMPLE

This section will show how the Monitor can be used to modify itself for the addition of a new command. The command will have the name AA, and the code for the command has been generated onto a disk file named CODE, assembled for execution beginning at 800H. The new version of the Monitor will be a version that runs at 0.

#LF M0000 4000	Load the standard version of the origin zero Monitor to RAM.
#LF CODE 4800	Load the code for the new Monitor command immediately after the copy of the Monitor in RAM.
#GO M0000	Load and execute the standard Monitor.
>SM 4000,800 0	Find the zero block for command table expansion. Locate the first block of seventeen zeros (assume for this example that the location of the first zero in the block is at address 4241H).
>FM 4241 41 >FM 4242 41 >FM 4244 08	These commands add the command name AA to the command table and set the start address of the code for the command to be 800H.
>OS	Return to the GDOS.
#CR NEWMON 10	Create a new file, two blocks larger than the standard Monitor, to contain the modified version of the Monitor.
#TY NEWMON 1 0	Set the type of the new Monitor file to 1 and set the go-address to 0.
#SF NEWMON 4000	Save the expanded new Monitor to disk.
#GO NEWMON	Load and execute the new Monitor
>AA	Test the new command.

The following table is intended to ease the task of conversion between the various numeric representations commonly used in programming, as well as between numbers (of any kind) and the ASCII character code.

Note that the ASCII character set only goes as far as decimal 127 (7FH, 01111111 B). Also, many "characters" in ASCII are non-printing CONTROL CHARACTERS. Whenever a code corresponds to a printable character, that will be given. In the case of control characters, a description or name for the special character will be given in parentheses.

DECIMAL	HEX	BINARY	ASCII
0	00H	00000000	(NUL)
1	01H	00000001	(CONTROL-A)
2	02H	00000010	(CONTROL-B)
3	03H	00000011	(CONTROL-C)
4	04H	00000100	(CONTROL-D)
5	05H	00000101	(CONTROL-E)
6	06H	00000110	(CONTROL-F)
7	07H	00000111	(CONTROL-G, RINGS BELL)
8	08H	00001000	(CONTROL-H, BACKSPACE)
9	09H	00001001	(CONTROL-I, TAB)
10	0AH	00001010	(CONTROL-J, LINEFEED)
11	0BH	00001011	(CONTROL-K)
12	0CH	00001100	(CONTROL-L, FORMFEED)
13	0DH	00001101	(CONTROL-M, CARRIAGE RETURN)
14	0EH	00001110	(CONTROL-N)
15	0FH	00001111	(CONTROL-O)
16	10H	00010000	(CONTROL-P)
17	11H	00010001	(CONTROL-Q)
18	12H	00010010	(CONTROL-R)
19	13H	00010011	(CONTROL-S)
20	14H	00010100	(CONTROL-T)
21	15H	00010101	(CONTROL-U)
22	16H	00010110	(CONTROL-V)
23	17H	00010111	(CONTROL-W)
24	18H	00011000	(CONTROL-X)
25	19H	00011001	(CONTROL-Y)
26	1AH	00011010	(CONTROL-Z)
27	1BH	00011011	(ESCAPE)
28	1CH	00011100	(NON-PRINTING)
29	1DH	00011101	(NON-PRINTING)
30	1EH	00011110	(NON-PRINTING)
31	1FH	00011111	(NON-PRINTING)

CONVERSION TABLE (continued)

DECIMAL	HEX	BINARY	ASCII
32	20H	00100000	(SPACE)
33	21H	00100001	!
34	22H	00100010	"
35	23H	00100011	#
36	24H	00100100	$
37	25H	00100101	%
38	26H	00100110	&
39	27H	00100111	'
40	28H	00101000	(
41	29H	00101001	)
42	2AH	00101010	*
43	2BH	00101011	+
44	2CH	00101100	,
45	2DH	00101101	-
46	2EH	00101110	.
47	2FH	00101111	/
48	30H	00110000	0
49	31H	00110001	1
50	32H	00110010	2
51	33H	00110011	3
52	34H	00110100	4
53	35H	00110101	5
54	36H	00110110	6
55	37H	00110111	7
56	38H	00111000	8
57	39H	00111001	9
58	3AH	00111010	:
59	3BH	00111011	;
60	3CH	00111100	<
61	3DH	00111101	=
62	3EH	00111110	>
63	3FH	00111111	?

CONVERSION TABLE (continued)

DECIMAL	HEX	BINARY	ASCII
64	40H	01000000	@
65	41H	01000001	A
66	42H	01000010	B
67	43H	01000011	C
68	44H	01000100	D
69	45H	01000101	E
70	46H	01000110	F
71	47H	01000111	G
72	48H	01001000	H
73	49H	01001001	I
74	4AH	01001010	J
75	4BH	01001011	K
76	4CH	01001100	L
77	4DH	01001101	M
78	4EH	01001110	N
79	4FH	01001111	O
80	50H	01010000	P
81	51H	01010001	Q
82	52H	01010010	R
83	53H	01010011	S
84	54H	01010100	T
85	55H	01010101	U
86	56H	01010110	V
87	57H	01010111	W
88	58H	01011000	X
89	59H	01011001	Y
90	5AH	01011010	Z
91	5BH	01011011	[
92	5CH	01011100	\
93	5DH	01011101	]
94	5EH	01011110	^B| OR ^
95	5FH	01011111	_

CONVERSION TABLE (continued)

DECIMAL	HEX	BINARY	ASCII
96	60H	01100000	`
97	61H	01100001	a
98	62H	01100010	b
99	63H	01100011	c
100	64H	01100100	d
101	65H	01100101	e
102	66H	01100110	f
103	67H	01100111	g
104	68H	01101000	h
105	69H	01101001	i
106	6AH	01101010	j
107	6BH	01101011	k
108	6CH	01101100	l
109	6DH	01101101	m
110	6EH	01101110	n
111	6FH	01101111	o
112	70H	01110000	p
113	71H	01110001	q
114	72H	01110010	r
115	73H	01110011	s
116	74H	01110100	t
117	75H	01110101	u
118	76H	01110110	v
119	77H	01110111	w
120	78H	01111000	x
121	79H	01111001	y
122	7AH	01111010	z
123	7BH	01111011	{
124	7CH	01111100	|
125	7DH	01111101	}
126	7EH	01111110	~
127	7FH	01111111	(DELETE, RUB OUT)

CONVERSION TABLE (continued)

DECIMAL	HEX	BINARY	ASCII
128	80H	10000000	
129	81H	10000001	
130	82H	10000010	
131	83H	10000011	
132	84H	10000100	
133	85H	10000101	
134	86H	10000110	
135	87H	10000111	
136	88H	10001000	
137	89H	10001001	
138	8AH	10001010	
139	8BH	10001011	
140	8CH	10001100	
141	8DH	10001101	
142	8EH	10001110	
143	8FH	10001111	
144	90H	10010000	
145	91H	10010001	
146	92H	10010010	
147	93H	10010011	
148	94H	10010100	
149	95H	10010101	
150	96H	10010110	
151	97H	10010111	
152	98H	10011000	
153	99H	10011001	
154	9AH	10011010	
155	9BH	10011011	
156	9CH	10011100	
157	9DH	10011101	
158	9EH	10011110	
159	9FH	10011111	

CONVERSION TABLE (continued)

DECIMAL	HEX	BINARY	ASCII
160	A0H	10100000	
161	A1H	10100001	
162	A2H	10100010	
163	A3H	10100011	
164	A4H	10100100	
165	A5H	10100101	
166	A6H	10100110	
167	A7H	10100111	
168	A8H	10101000	
169	A9H	10101001	
170	AAH	10101010	
171	ABH	10101011	
172	ACH	10101100	
173	ADH	10101101	
174	AEH	10101110	
175	AFH	10101111	
176	B0H	10110000	
177	B1H	10110001	
178	B2H	10110010	
179	B3H	10110011	
180	B4H	10110100	
181	B5H	10110101	
182	B6H	10110110	
183	B7H	10110111	
184	B8H	10111000	
185	B9H	10111001	
186	BAH	10111010	
187	BBH	10111011	
188	BCH	10111100	
189	BDH	10111101	
190	BEH	10111110	
191	BFH	10111111	

CONVERSION TABLE (continued)

DECIMAL	HEX	BINARY	ASCII
192	C0H	11000000	
193	C1H	11000001	
194	C2H	11000010	
195	C3H	11000011	
196	C4H	11000100	
197	C5H	11000101	
198	C6H	11000110	
199	C7H	11000111	
200	C8H	11001000	
201	C9H	11001001	
202	CAH	11001010	
203	CBH	11001011	
204	CCH	11001100	
205	CDH	11001101	
206	CEH	11001110	
207	CFH	11001111	
208	D0H	11010000	
209	D1H	11010001	
210	D2H	11010010	
211	D3H	11010011	
212	D4H	11010100	
213	D5H	11010101	
214	D6H	11010110	
215	D7H	11010111	
216	D8H	11011000	
217	D9H	11011001	
218	DAH	11011010	
219	DBH	11011011	
220	DCH	11011100	
221	DDH	11011101	
222	DEH	11011110	
223	DFH	11011111	

CONVERSION TABLE (continued)

DECIMAL	HEX	BINARY	ASCII
224	E0H	11100000	
225	E1H	11100001	
226	E2H	11100010	
227	E3H	11100011	
228	E4H	11100100	
229	E5H	11100101	
230	E6H	11100110	
231	E7H	11100111	
232	E8H	11101000	
233	E9H	11101001	
234	EAH	11101010	
235	EBH	11101011	
236	ECH	11101100	
237	EDH	11101101	
238	EEH	11101110	
239	EFH	11101111	
240	F0H	11110000	
241	F1H	11110001	
242	F2H	11110010	
243	F3H	11110011	
244	F4H	11110100	
245	F5H	11110101	
246	F6H	11110110	
247	F7H	11110111	
248	F8H	11111000	
249	F9H	11111001	
250	FAH	11111010	
251	FBH	11111011	
252	FCH	11111100	
253	FDH	11111101	
254	FEH	11111110	
255	FFH	11111111	

FUNCTION	CHARACTERS	DECIMAL CODE	HEX CODE
AUTO-WRAPAROUND OFF	ESC $	27 36	1BH 25H
AUTO-WRAPAROUND ON	ESC %	27 37	1BH 24H
BACKSPACE	CONTROL-H	8	08H
BELL	CONTROL-G	7	07H
CARRIAGE RETURN	CONTROL-M	13	0DH
CLEAR TO END OF LINE	CONTROL-N	14	0EH
CLEAR TO END OF SCREEN	CONTROL-O	15	0FH
CURSOR OFF	CONTROL-Y	25	19H
CURSOR ON	CONTROL-X	24	18H
DESTRUCTIVE BACKSPACE	DELETE	127	7FH
DIRECT CURSOR ADDRESSING	ESC =	26 61	1BH 3DH
	X Y	X Y	X Y
FORESPACE	CONTROL-L	12	0CH
HOME CURSOR	CONTROL-^	30	1EH
HYBRID MODE OFF	CONTROL-S	19	13H
HYBRID MODE ON	CONTROL-R	18	12H
KEYBOARD AUDIO FEEDBACK OFF	CONTROL-F	6	06H
KEYBOARD AUDIO FEEDBACK ON	CONTROL-E	5	05H
LINEFEED	CONTROL-J	10	0AH
NEW LINE	CONTROL-_	31	1FH
REVERSE LINEFEED	CONTROL-K	11	0BH
REVERSE VIDEO OFF	CONTROL-B	2	02H
REVERSE VIDEO ON	CONTROL-A	1	01H
SCREEN OFF	CONTROL-]	29	1DH
SCREEN ON	CONTROL-\	28	1CH
SMOOTH SCROLLING OFF	CONTROL-U	21	15H
SMOOTH SCROLLING ON - FAST	CONTROL-Q	17	11H
SMOOTH SCROLLING ON - SLOW	CONTROL-T	20	14H
VIDEO RESET	CONTROL-D	4	04H

ASCII CHARACTER	DECIMAL VALUE	HEXADECIMAL VALUE	LINE NUMBER	COLUMN NUMBER
SPACE	32	20H	X= 1	Y= 1
!	33	21H	2	2
"	34	22H	3	3
#	35	23H	4	4
$	36	24H	5	5
%	37	25H	6	6
&	38	26H	7	7
'	39	27H	8	8
(	40	28H	9	9
)	41	29H	10	10
*	42	2AH	11	11
+	43	2BH	12	12
,	44	2CH	13	13
-	45	2DH	14	14
/	47	2FH	16	16
0	48	30H	17	17
1	49	31H	18	18
2	50	32H	19	19
3	51	33H	20	20
4	52	34H	21	21
5	53	35H	22	22
6	54	36H	23	23
7	55	37H	24	24
8	56	38H		25
9	57	39H		26
:	58	3AH		27
;	59	3BH		28
<	60	3CH		29
=	61	3DH		30
>	62	3EH		31
?	63	3FH		32
@	64	40H		33
A	65	41H		34
B	66	42H		35
C	67	43H		36
D	68	44H		37
E	69	45H		38
F	70	46H		39
G	71	47H		40

VALUES FOR DIRECT CURSOR ADDRESSING (continued)

ASCII CHARACTER	DECIMAL VALUE	HEXADECIMAL VALUE	LINE NUMBER	COLUMN NUMBER
H	72	48H		41
I	73	49H		42
J	74	4AH		43
K	75	4BH		44
L	76	4CH		45
M	77	4DH		46
N	78	4EH		47
O	79	4FH		48
P	80	50H		49
Q	81	51H		50
R	82	52H		51
S	83	53H		52
T	84	54H		53
U	85	55H		54
V	86	56H		55
W	87	57H		56
X	88	58H		57
Y	89	59H		58
Z	90	5AH		59
[	91	5BH		60
\	92	5CH		61
]	93	5DH		62
^	94	5EH		63
_	95	5FH		64
`	96	60H		65
a	97	61H		66
b	98	62H		67
c	99	63H		68
d	100	64H		69
e	101	65H		70
f	102	66H		71
g	103	67H		72
h	104	68H		73
i	105	69H		74
j	106	6AH		75
k	107	6BH		76
l	108	6CH		77
m	109	6DH		78
n	110	6EH		79
o	111	6Fh		80

The following table shows how the 64K byte RAM address space is allocated for the standard version system software and hardware. All addresses are given in hexadecimal notation.

LOAD ADDRESS	PROGRAMS
E000H-FFFFH	GDOS
0000H-DFFFH *	BASIC, GBASIC,
0000H-07FFH	M0000
C000H-C7FFH	MC000
0000H-VARIOUS	Utilities CO, CD, CF, DT, CK
0000H-DFFFH	USER PROGRAMS

* The upper limit of BASIC can be set by the user with the MEMSET command. It is initially set to DFFF.